View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

Key Lengths

Contribution to The Handbook of Information Security

Arjen K. Lenstra

Citibank, N.A., and Technische Universiteit Eindhoven
1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.
arjen.lenstra@citigroup.com

Abstract. The key length used for a cryptographic protocol determines the highest security it can offer. If the key is found or 'broken', the security is undermined. Thus, key lengths must be chosen in accordance with the desired security. In practice, key lengths are mostly determined by standards, legacy system compatibility issues, and vendors. From a theoretical point of view selecting key lengths is more involved. Understanding the relation between security and key lengths and the impact of anticipated and unexpected cryptanalytic progress, requires insight into the design of the cryptographic methods and the mathematics involved in the attempts at breaking them. In this chapter practical and theoretical aspects of key size selection are discussed.

1 Introduction

In cryptographic context, 40, 56, 64, 80, 90, 112, 128, 155, 160, 192, 256, 384, 512, 768, 1024, 1536, 2048, and 4096 are examples of key lengths. What they mean and how they are and should be selected is the subject of this chapter.

Key lengths indicate the number of bits contained in a certain cryptographic key or related arithmetic structure. They are a measure for the security that may be attained. To the uninitiated, however, the relation between key lengths and security is confusing. To illustrate, key lengths 80, 160, and 1024, though quite different, may imply comparable security when 80 is the key length for a symmetric encryption method, 160 a hash length, and 1024 the bit length of an RSA modulus. Part of this correspondence follows immediately from the well known 'fact' that symmetric encryption with B-bit keys and $2B$-bit cryptographic hashes offer the 'same' security. But the correspondence with 1024-bit RSA is quite a different story that allows many variations. In the sequel an attempt is made to view this and other key length issues from all reasonable perspectives.

Key lengths are often powers of 2 or small multiples thereof. This is not for any mathematical or security reason. It is simply because data

are usually most conveniently processed and stored in chunks of 8 bits (bytes), 32 bits (words), 64 bits (blocks), etc.

Symmetric encryption and cryptographic hashing. Ideally, the long term prospects of the relationship between key length and security should be well understood when key length decisions are made. In case of symmetric encryption and cryptographic hashing methods the decision is facilitated for most users by the following three facts:

- There is broad consensus which key lengths are 'conservative', i.e., have good prospects to offer very long term security.
- Nowadays, most default choices available on the marketplace are conservative.
- The performance is barely, if at all, affected by the key length choice.

Thus, for symmetric encryption and cryptographic hashing it suffices to make a reasonably well informed conservative choice.

Asymmetric systems. As hinted at above, there is much less agreement about conservative choices for asymmetric systems such as RSA. Furthermore, for these systems the performance *does* deteriorate with increasing key lengths. Even if a consensual conservative choice could be made, it may not be a choice that is practically feasible. In practice most users of asymmetric systems follow the recommendations of the vendor community. But there is no guarantee that the vendor community always has sufficient business incentive to comply with the recommendations of the standards bodies, or that the latter fully understand all relevant issues. The main purpose of the present chapter is to offer unbiased advice to the more prudent users of asymmetric systems to help decide which of the available options may be adequate for their purposes.

Security in practice. The security that corresponds to a key length choice for a cryptographic protocol represents the highest security that can, in principle, be achieved by the system incorporating that protocol. Systems are usually most efficiently attacked by exploiting other than cryptographic key related weaknesses. Examples are imperfections in the underlying protocol, in the implementation, the environment, or the users. Selecting appropriate key lengths may therefore be regarded as an academic exercise. It should be kept in mind, however, that inadequate key length choices do affect the security of a system. In the remainder of this chapter security-affecting issues other than key lengths are not further discussed.

Overview. This chapter is organized as follows. Section 2 introduces the concept of security level and contains the general background for

the remainder of the chapter. Key lengths for symmetric systems are discussed in Section 3 and cryptographic hash function sizes in Section 4. An overview of asymmetric methods is given in Section 5, which leads to the discussion of factoring based systems in Section 6 and of discrete logarithm based systems in Section 7. The reader who is not familiar with common cryptographic concepts such as symmetric and asymmetric cryptosystems may look them up in other chapters to this handbook.

In the sequel, $\log x$ denotes the natural logarithm of x and $\log_b x$ denotes the base b logarithm of x. As customary, $\exp(x) = e^x$.

2 Security Level

General attacks. In this section *general attacks* are considered. For symmetric systems these are attacks where the key has to be recovered given a single known (plaintext, ciphertext) pair. For block ciphers the plaintext consists of a single or at most a few blocks. It is assumed that this pair uniquely determines the key, or that correctness of the key can independently be verified. Refer to [6] for a discussion of the latter point. For asymmetric systems general attacks are attacks where the private key has to be found given the public key. General attacks most closely correspond to real life situations. They exclude attacks where the attacker has access to any other data that can be generated only by means of the unknown key, such as in differential and linear cryptanalysis of block ciphers.

Security level. If a symmetric cryptosystem with λ-bit keys does not allow a general attack that is faster than exhaustive key search, then it is traditionally said to have *security level* λ. Exhaustive key search for λ-bit keys may involve up to 2^λ different keys. In general, a cryptographic system offers security level λ if a successful general attack can be expected to require effort approximately 2^λ.

Relation between security level and security. To determine if a cryptographic system offers adequate security or protection, it is not immediately useful to tie the definition of security level to symmetric cryptosystem key lengths. In the first place, the amount of time and money required to realize an attack effort decreases over time because computers become faster and cheaper. Thus, the amount of protection offered by a certain fixed security level is constantly eroded. A related point is that cryptanalytic progress over time may affect the security level of a cryptographic system not by lowering the cost to realize a certain attack, but by proposing an improved attack method. Furthermore, the definition of se-

curity level involves an unspecified constant of proportionality—vaguely indicated by the 'approximately 2^λ'—and thus its meaning may vary from system to system. And finally, 'adequate protection' is a vague term whose interpretation depends on the application one has in mind, and is even then still subjective. In the remainder of this section these issues affecting the relation between security levels and security are addressed, which allows selection of key lengths corresponding to any amount of protection one feels comfortable with.

Modelling the relation. Although a cryptosystem's security level may not be indicative of its effectiveness, security levels allow comparison of the security offered by systems. Assuming identical constants of proportionality and environments, a system of security level $\lambda + \mu$ may be expected to be 2^μ times harder to attack, and thus be 2^μ times more secure, than a system of security level λ. Once it has been agreed that a certain security level offers an adequate amount of protection in a certain known (past) environment, twice the protection can be achieved in that environment by incrementing the security level by one (assuming other characteristics of the system involved are not affected by the change). And, more in general, an x times higher amount of protection follows, in that same environment, by adding $\log_2 x$ to the security level. If, additionally, the effect of changes in the environment is modelled, then a more general correspondence can be derived between security levels and amount of protection for any (future) environment. As indicated above, these environmental changes come in two flavors: changes in the computational environment that affect the amount of protection by lowering the cost at which the same attack can be realized but that leave the security level itself unchanged, and changes in the cryptanalytic environment that allow a different type of attack thereby lowering the security level. The presentation below heavily relies on [28] where this approach was first proposed.

Defining adequate protection. The Data Encryption Standard (DES) is a symmetric cryptosystem with 56-bit keys, published in 1977 by the U.S. Department of Commerce [32] and supposed to be reviewed once every five years. There was some skepticism about the security level of the DES. But despite extensive cryptanalysis no better general attack than exhaustive key search has been found and the security level is generally believed to be 56.

Because the DES was widely adopted, there must have been broad consensus that in 1982, the first year the DES would come up for review, it offered an adequate amount of protection for commercial applications.

For that reason, and for the purposes of this chapter, *adequate protection* is defined as the security offered in 1982 by the DES. Disregarding the effect of the constant of proportionality, this is synonymous with *security level 56 in 1982*. In the remainder of this section it is discussed what security level can be expected to offer adequate protection until the year of one's choice. It is left to the reader to determine how the definition of adequate protection compares to one's own security requirements and, if desired, to change the default choice made above. The paragraphs below may be helpful for this purpose.

The cost of breaking the DES. To put the definition of adequate protection in a different light, in 1980 it was estimated that, in 1980 money and technology, an exhaustive key search attack against the DES would require on average 2 days on a device that would cost approximately US$50 million to build. The design underlying this estimate is fully parallelizable in the sense that a w times slower (or faster) device would be w times cheaper (or more expensive), for any reasonable w. As first suggested in this context in [2], this implies that an appropriate way to measure an attack effort is obtained by multiplying the time required by the equipment cost; see also [26] and [49] where this measure is referred to as the *throughput cost* and *full cost*, respectively. Below it is simply referred to as the *cost* of an attack effort. This cost will be measured in *dollardays*: in 1980 the DES could be broken in approximately 100M dollardays. The cost does **not** include the one time overhead for the detailed design specifications.

Modelling the effect of changes in the computational environment. Technical progress had a profound effect on the security of the DES. In 1993 a DES key search engine was proposed that would require about 150K dollardays, down from the 100M dollardays required by the 1980 design [48]. And in 1998 a parallel hardware device was built for US$130K including design overhead, and used to crack the DES in a matter of days [11, 19]. Thus, though security level 56 may have offered adequate protection for commercial applications in 1982, this is no longer the case in 2004.

The effect of changes in the computational environment is modelled using Moore's law. Traditionally, it says that the computing power per chip doubles every 18 months. To make Moore's law less technology dependent the following variant is adopted for this chapter:

Moore's law. The cost of any fixed attack effort drops by a factor 2 every 18 months.

This can be seen to be in reasonable correspondence with the various DES cracking devices referred to above. It is also an acceptable compromise between those who argue that this rate of progress cannot be sustained and those who find it prudent to expect more rapid progress. It follows that the 100M dollardays cost of the 1980 DES cracker would be reduced to 40M dollardays in 1982, because $40 \approx 100/2^{24/18}$.

Obviously, all estimates of this sort based on Moore's law have to be taken with a grain of salt and interpreted appropriately: the approximate values and growth rates matter, not the precise figures.

The cost of adequate protection. Irrespective of the speed or type of the cryptosystem, a cryptosystem is said to offer adequate protection until a given year if the cost of a successful attack measured in that year can be expected to be approximately 40M dollardays. See below how to change the cost figure corresponding to adequate protection from 40M to $x * 40M$ if 40M is felt to be inadequate ($x > 1$) or overkill ($0 < x < 1$). For reasonable values of x the effect of the resulting corrections is mostly negligible, since only the approximate values matter.

For asymmetric systems based on the factoring problem or the general problem of computing discrete logarithms in multiplicative groups of finite fields the 40M dollardays cost measure will be used to determine adequate protection. For other asymmetric systems based on the discrete logarithm problem, symmetric systems, and cryptographic hash functions one can instead use the approach based on security levels combined with Moore's law. To allow comparison with DES security levels the effect of the constant of proportionality must be taken into account, at least in principle. Below it is shown how this is done.

The effect of Moore's law. It follows from Moore's law that to maintain the same amount of protection once every 18 months the security level should be incremented by one, assuming the speed is not affected. Thus, assuming the same speed as the DES, a symmetric system of security level $56 + 10 = 66$ would offer adequate protection in 1997, since $1997 - 1982 = 15$ years covers 10 periods of 18 months. Under the same assumption, security levels 76 and 86 should be adequate until 2012 and 2027, respectively.

More in general, a symmetric system of speed comparable to the DES would offer adequate protection until the year $y = 1982+15x$ if its security level is $\lambda = 56 + 10x$. Given a year security level λ, the year $y(\lambda)$ until which it offers adequate protection is thus calculated as

$$y(\lambda) = 1982 + \frac{3(\lambda - 56)}{2}. \tag{1}$$

Conversely, given a year y, the security level $\lambda(y)$ that offers adequate protection until year y is

$$\lambda(y) = 56 + \frac{2(y - 1982)}{3}. \tag{2}$$

Although this may be a reasonable model that leads to a useful computational tool, it would stretch the imagination to use it beyond, say, the year 2050. But it is, for instance, not unreasonable to conclude that the widely used security level $\lambda = 80$ offers adequate protection until the year

$$y(80) = 1982 + \frac{3(80 - 56)}{2} = 2018$$

(cf. equation (1)).

The effect of the constant of proportionality. If a symmetric system is $s > 0$ times faster than the DES, exhaustive key search and thus general attacks are s times faster as well. To compensate, either for the same year $\log_2 s$ should in principle be added to the security level, or for the same security level $1.5 \log_2 s$ has to be subtracted from the year. Ciphers faster than the DES ($s > 1$) require a higher security level, or the same security level does not last as long. But for slower ciphers ($s < 1$) a lower security level suffices, or the same security level lasts longer.

In theory this correction based on the speed compared to the DES takes care of the unspecified constant of proportionality mentioned above. In practice, however, the effect of this correction is mostly negligible. Not only is $|\log_2 s|$ typically small, but also making such corrections would lead to a misleading sense of precision contradictory to the way these estimates should be interpreted.

Alternative definitions of adequate protection. Defining adequate protection as the security offered by security level 56 in 1982 may be a reasonable compromise. But it is a subjective choice. If 'security level 56 in year Y' better reflects one's feelings, then one should replace in the sequel all occurrences of '1982' by Y. Furthermore, in the '40M dollar-days cost' associated with adequate protection, the '40' must be divided by $2^{2(Y-1982)/3}$. For instance, if the DES was still felt to offer adequate protection in the year 1990, replace 1982 by 1990 throughout, and '40M' by $1M$, since $2^{2(1990-1982)/3} \approx 40$.

Similarly, if one is more comfortable with interpretation of cost figures and finds the '40M dollardays' inappropriate, replace the '40' in the sequel by $x * 40$ for any $x \neq 1$ of one's choice. As a consequence, all occurrences of the year 1982 must be replaced by $1982 - 1.5 \log_2(x)$.

Modelling the effect of changes in cryptanalytic capabilities. Moore's law may act as a self-fulfilling prophesy by influencing and controlling the development of the steady stream of improvements required to sustain it. There is no similar mechanism controlling the rate of cryptanalytic progress.

Moore's law affects all cryptosystems across the board in the same way by lowering the cost of attacks. Cryptanalytic progress, on the other hand, usually affects the security level of one particular type of cryptosystem while leaving that of others untouched. An advance in factoring does not affect the security level of symmetric cryptosystems, and a newly found peculiarity in the design of an S-box used by some symmetric cryptosystem has no effect on the security level of RSA or of symmetric systems using non-affected designs. Furthermore, the overall effect of cryptanalytic progress may vary from system to system. When a new weakness in a symmetric system or cryptographic hash function is discovered, it may be possible to modify or simply retire it, because relatively small modifications often render new attacks useless and, if not, there are enough equivalent alternative systems and functions to choose from. In the asymmetric case the situation is different. The luxury of a quick switch to an alternative system can generally not be afforded because there are not that many different equivalent schemes. As a result, adapting key lengths may be the only option to compensate for the effects of a new cryptanalytic insight such as a new algorithm to solve the mathematical problem underlying an asymmetric system.

There are cryptographic applications, however, where system modification or retirement and key length adaptations are not feasible, and where adequate protection must be maintained for an extended period of time, even in the presence of cryptanalytic progress discovered after the application was put to use. For instance, in long term confidential data storage in an environment that is not necessarily adequately protected, the fixed stored data must remain undecipherable as long as the confidentiality must last. With the present state of the art of cryptology disasters can always happen, and adequate long term protection cannot be fully guaranteed. Barring disastrous cryptanalytic progress, however, proper application of suitably modelled cryptanalytic progress leads to an acceptable practical solution for long term protection as well.

It remains to model cryptanalytic progress. A priori it is unclear how this should be done. However, since there is no reason to expect significant changes in the global research community that is interested in cryptanalysis, it is assumed that the rate of cryptanalytic progress in

the future is the same as it was since cryptography became more of a mainstream public-domain activity. Because past cryptanalytic progress varied considerably between different cryptographic systems, a specific cryptanalytic progress model is defined for each of the various systems. The details of each model are described in the relevant sections below.

3 Symmetric cryptosystems

Symmetric cryptosystems are encryption methods where sender and receiver share a key for encryption and decryption, respectively. Examples are block and stream ciphers. There is a great variety of such systems, but only a few of them are generally accepted and widely used. The popular block ciphers, with the exception of the original DES, can be expected to offer adequate protection (cf. Section 2) for the foreseeable future. Thus, from a pragmatic point of view, key length selection for block ciphers is hardly an issue as long as one sticks to widely used modern schemes. In this section some issues are discussed concerning security levels and key lengths for a number of popular block ciphers.

Stream ciphers are more problematic. They are not considered here for a variety of reasons. Often their design is proprietary or their usage subject to licensing restrictions. Their cryptanalysis is too much in a state of flux and their security level influenced by the way they are used. For instance, the strong version A5/1 of a stream cipher used in the European cellphone industry can trivially be broken [4], a similar application of the stream cipher RC4 was found to be completely insecure [12], and the stream cipher SEAL has been revised several times [15]. Finally, all six stream ciphers submitted to the NESSIE initiative [33] were found to be too weak and none was selected, illustrating the apparent difficulty of designing stream ciphers.

Block ciphers. Table 1 lists some common block ciphers along with their key length choices, block lengths, and the most up-to-date information about their security levels under general attacks. The list is for illustrative purposes only and is not, nor is it meant to be, exhaustive. In- or exclusion of a cipher in no means indicates the author's support for that cipher or lack thereof. Although other types of attacks are not considered to determine the security level (cf. Section 2), the more recent block ciphers are designed to have strong resistance against those attacks as well.

For triple DES the lower security levels are of mostly theoretical significance as they assume that memory costs are ignored, but the higher ones are more realistic. For DESX (cf. [17]) and IDEA (cf. [3]) the secu-

rity levels in Table 1 are based on the fact that even after many years no effective cryptanalysis has been published, as far as general attacks are concerned. For the Advanced Encryption Standard (AES) they may be based on wishful thinking because at the time of writing of this chapter the AES has been scrutinized for only a few years. But this is combined with the expectation (based on the sudden replacement of SHA by SHA-1, see Section 4) that if anytime soon something serious affecting the AES would be found, a modification would be introduced.

Table 1. Common block ciphers.

name	key length	block length	security level
DES	56	64	56
two key triple DES	112	64	95-100
three key triple DES	168	64	112-116
DESX	120	64	120
IDEA	128	64	128
AES-128	128	128	128
AES-192	192	128	192
AES-256	256	128	256

Cryptanalytic developments. Table 1 illustrates two points. In the first place, two and three key triple DES are the only ciphers in the table for which the security level is smaller than the key length. This is the case for multiple encryption in general (cf. [49]). Nevertheless, despite the fact that their security level can never equal their key length, multiple encryption methods may be a convenient way to boost security by repeated application of an available cipher when replacement by a stronger one is not an option. Secondly, Table 1 shows that, other than for legacy reasons, there is in principle no reason to settle for a cipher that offers a security level lower than its key length.

Performance considerations. As indicated in Section 2 a proper interpretation and comparison of the security levels in Table 1 in principle requires knowledge of the relative speeds of the various block ciphers. It is also mentioned, however, that this type of 'overprecision' has no practical relevance. This is illustrated here.

According to equation (1) from Section 2, security level λ offers adequate protection until the year $y(\lambda) = 1982 + \frac{3(\lambda-56)}{2}$, disregarding the effect caused by the speed relative to the DES. A block cipher of security level $\lambda \geq 128$ leads to an un-corrected year estimate of $y(128) = 2090$ and beyond. Proper interpretation of this result is that security level 128 should suffice for, say, the next three decades and probably even longer.

Incorporation of the effect of the speed compared to the DES has no effect. For instance, IDEA and the DES have comparable hardware performance but in software IDEA is approximately twice faster (i.e., $s = 2$ in the notation of Section 2). So, in principle it would be 'correct', and may be even believed to be prudent, to subtract $1.5 \log_2 2 = 1.5$ from the year 2090, as set forth in Section 2. But the practical conclusion that IDEA should offer adequate security for the foreseeable future remains untouched by this correction. The same practical conclusion would be reached for block ciphers of security level 128 that would be a million times faster or slower than the DES.

Other considerations. Another issue with block ciphers is their block length. With b-bit blocks, and under reasonable assumptions regarding randomness of the inputs and the cipher's output behavior, a duplicate output block may be expected after about $2^{b/2}$ blocks have been encrypted. A duplicate generated with the same key may facilitate cryptanalysis and should be avoided.

When $b = 64$, this implies that the key should be refreshed well before 2^{32} blocks of 64-bits (i.e., 32 Gigabytes) have been encrypted—say after 10 gigabytes. When $b = 128$, the likelihood is negligible that duplicate blocks are encountered for any realistic amount of data properly encrypted with the same key.

Symmetric key lengths that offer adequate protection. With the exception of the DES, all ciphers listed in Table 1 offer adequate protection with respect to general attacks at least until the year 2030: even the weakest among them, two key triple DES, may be expected to offer adequate security until 2040 since, according to equation (1) in Section 2, $y(95) = 1982 + \frac{3(95-56)}{2} = 2040.5$ (with, if $s = 1/3$ is thought to reflect its performance degradation compared to the DES, an overzealous correction to the year $2040.5 - \log_2 1/3 \approx 2042$). Furthermore, given the virtual lack of cryptanalytic progress with respect to general attacks and assuming current cryptanalytic trends persist (i.e., that cryptanalysis remains relative ineffective), the ciphers of security level ≥ 128 can be expected to offer adequate protection for any conceivable commercial application, including long term data storage, and as long as anyone can reasonably predict. Thus, most ciphers from Table 1 with the exception of the DES can safely be recommended, as long as the amount of data that will be encrypted with a single key is limited. If the latter cannot be guaranteed, the AES should be used.

In [5], which dates back from 1996, it is recommended that for adequate protection for the next 20 years, i.e., until the year 2016, keys in

newly-deployed symmetric cryptosystems should be at least 90 bits long. According to the estimates presented here, security level $\lambda = 90$ would offer adequate security until the year $y(90) = 1982 + \frac{3(90-56)}{2} = 2033$ and security level $\lambda(2016) = 56 + \frac{2(2016-1982)}{3} = 78\frac{2}{3}$ would suffice until the year $y = 2016$ (cf. equations (1) and (2) in Section 2). Thus, the recommendation of [5] is conservative and can be followed without hesitation.

It may seem wasteful to use a key length such as 128 that leads to a security level that is so much larger than necessary. As far as the speed of the cryptosystem is concerned, this is not an issue because the key size does not have a major impact on the speed. If the 'overlong' key is problematic because of other concerns such as cost of key exchange or storage, a sufficiently shortened but still adequately long version may be used and padded with a fixed sequence of bits known as *salt*. This recommendation should not be followed blindly in case of triple DES, unless one knows what one is doing.

4 Cryptographic hash functions

Given an input consisting of an arbitrary sequence of bits, a cryptographic hash function efficiently produces a fixed length output, the *hash* of the input. In this section H denotes the bit-length of the hash. The output is intended as a 'fingerprint' of the input in data integrity and authentication applications. Therefore, cryptographic hash functions must have a number of properties that make them suitable for these applications. In the first place given any output value for which the corresponding input is unknown, it must be computationally infeasible to find any input that hashes to that output. Secondly, for a known (input, output) pair, it must be computationally infeasible to find another input that hashes to the same output. Although these two properties suffice for many applications (cf. [1]) it is common to assume a stronger version of the last property, namely that it must be computationally infeasible to find two distinct inputs that hash to the same output. This last requirement is often referred to as *collision resistance*.

The issue at discussion here are the requirements on H without which a cryptographic hash function cannot have the desired properties, i.e., the length requirements that must be met irrespective of any of the other properties of the hash function. Obviously, satisfying the requirements on H does not guarantee proper design of the hash function, it is just a necessary first step.

Assume that the output of a hash function behaves as a uniformly distributed random H-bit value. It follows from the first two requirements that H must be chosen such that it is computationally infeasible to perform 2^H applications of the hash function (for random inputs). Thus, to achieve security level λ and to satisfy the first two requirements, it must be the case that $H \geq \lambda$.

The collision resistance requirement, however, has more severe consequences for H. If values are drawn at random from a set of cardinality C then the expected number of draws before an element is drawn twice (a so-called *collision*) is approximately $1.25\sqrt{C}$. This fact is commonly known as the *birthday paradox*. If follows that if the hash is computed of different randomly selected inputs, a duplicate output can be expected after about $1.25 * 2^{H/2}$ attempts. To achieve security level λ and to satisfy the third requirement, it must therefore be the case that $H \geq 2\lambda$.

The search for a collision as described above is commonly known as a *collision attack*. Resistance against exhaustive key search and collision attacks play comparable roles in the contexts of symmetric cryptosystems and cryptographic hash functions, respectively: well-designed symmetric systems do not allow general attacks faster than exhaustive key search, and well-designed cryptographic hash functions do not allow discovery faster than by collision attacks of a distinct pair of inputs with identical outputs.

Cryptographic hash functions. Table 2 lists some common hash functions along with their output lengths and the most up-to-date information about their security levels under collision attacks.

Table 2. Common cryptographic hash functions.

name	H	security level
RIPEMD-160	160	80
SHA-1	160	80
SHA-256	256	128
SHA-384	384	192
SHA-512	512	256

Cryptanalytic developments. Well-known precursors of the cryptographic hash functions in Table 2 are MD4, MD5, and RIPEMD-128, all with $H = 128$, and SHA, with $H = 160$. Significant deficiencies were found in their design. MD4 is considered to be broken and it is widely suspected that the security levels of MD5 and RIPEMD-128 are both lower than 64. Furthermore, a sufficiently serious problem was found in SHA to

replace it by SHA-1. For a discussion of these developments see [10], [38], and also [8].

The results of those cryptanalytic findings were incorporated in the design of the cryptographic hash functions in Table 2. That is no guarantee that those functions do not allow faster attacks than collision attacks. But it indicates that the functions from Table 2 were designed with a great deal of care and that an unanticipated new weakness most likely requires new cryptanalytic insights. Given how infrequently such insights occur, it is reasonable at this point to assume that the security levels in Table 2 are accurate for the foreseeable future. This should be combined with a conservative choice of cryptographic hash function and, where possible, application of the methods from [1] to design one's protocols in such a way that the cryptographic hash function does not have to be collision resistant, i.e., does not have to meet the third requirement. If the latter is properly done it effectively doubles the security level.

Performance considerations. Whether or not a cryptographic hash function of hash length H offers adequate protection until a certain year, as defined in Section 2, in principle depends on the relatively speed of the hash function compared to the DES. With inputs of comparable length, the speed of all common cryptographic hash function is comparable to the speed of common blockciphers, such as the DES. Thus, the effect of incorporating the speed is negligible to begin with. Furthermore, as argued in Section 3, for the larger H values the effect is best neglected anyhow because it would lead to inappropriately accurate interpretation of inherently inaccurate figures.

Cryptographic hash lengths that offer adequate protection. In combination with the findings of Sections 2 and 3 it follows that arbitrary application of cryptographic hash functions of security level λ (i.e., with $H = 2\lambda$ unless weaknesses exist) offers adequate protection until the year $y(\lambda) = 1982 + \frac{3(\lambda-56)}{2}$ (cf. equation (1) in Section 2). More in particular, the above cryptographic hash functions with $H = 160$, assuming they remain unbroken, may be expected to offer adequate protection until the year $y(160/2) = 1982$. All functions listed in Table 2 can be expected to offer adequate protection at least until the year 2030, very conservatively estimated, under the proviso that the functions of security level 80 are used in combination with the methods from [1]. As a rule of thumb, hash lengths must be chosen twice longer than symmetric key lengths.

5 Asymmetric methods

Private key and public key. In asymmetric cryptosystems each user, say A, has its own pair of keys: A's private key s_A and the corresponding public key p_A. Typically, the public key p_A can be used by any party to encrypt information intended for user A, which can then be decrypted by A using s_A. Alternatively, A may use s_A to digitally sign documents, and any party can use p_A to verify the resulting digital signatures. For some systems a single private/public key pair allows both en-/decryption and digital signatures, but great care has to be taken when doing so.

Performance deterioration. For symmetric cryptosystems and cryptographic hash functions the number of realistic alternatives is fairly limited, and their speed hardly depends on the key or hash length one settles for. For asymmetric cryptosystems the situation is different. There the performance of both the public operation (encryption or signature verification) and the private one (decryption or signature generation) deteriorates markedly, and possibly to different degrees, as the security level increases. Therefore, for asymmetric cryptosystems it is more important than for symmetric cryptosystems and cryptographic hash functions to determine the smallest key lengths that still offers the right amount of protection, thereby balancing security and performance requirements.

The design of asymmetric cryptosystems. The design of all common symmetric cryptosystems and cryptographic hash functions is mostly based on a combination of hard-to-define ingredients such as experience, avoidance of common errors, incorporation of the latest cryptanalytic insights, taste, sound judgment, and luck. As argued in [20], the design of the AES is a first attempt to a more scientific, less artful approach to block cipher design. All common asymmetric cryptosystems, on the other hand, are based on a well-defined mathematical problem, if at all possible combined with a proof that solving the latter is equivalent to breaking the system. The security of an asymmetric system is then based on the hope and belief that the mathematical problem does not allow an efficient solution. Sometimes that hope turns out to be ill-founded. For instance, the once popular knapsack-based public key cryptosystems were found to be susceptible to attacks using lattice basis reduction. Efficient lattice basis reduction methods thus meant the end for knapsack-based asymmetric cryptosystems.

Factoring and discrete logarithms. The two mathematical problems underlying the popular and by now 'classical' asymmetric cryptosystems are integer factorization and computing discrete logarithms, as described

below. Both these problems have been the subject of active research during the last few decades. Also, the cryptographic protocols they are embedded in have been widely studied, often resulting in provable equivalence of breaking the protocol and solving the mathematical problem. It turns out that the cryptanalytic progress, or lack thereof, affecting the corresponding asymmetric cryptosystems displays a smooth pattern without jumps or unwelcome 'surprises'. Under the assumption that this same smooth pattern persists, this allows reasonably well-founded analyses of key lengths required for adequate protection in the future. These analyses are presented in the subsequent sections.

It should be understood, however, that a clearly discernable and well-established past pattern in cryptanalytic progress is no guarantee that the future pattern will be the same or that there will not be any surprising breakthroughs. With the present state of the art there is no hard proof of the security of any of the popular asymmetric systems, simply because there are no proofs yet of the difficulty of any of the underlying mathematical problems: the only evidence of their difficulty is our failure to solve them. This is independent of any proofs of equivalence between a cryptosystem and its underlying mathematical problem. To refer to this provable equivalence as 'provable security', as common in the cryptographic literature, may be misleading, since what it actually means is 'provable equivalence to a problem of unproved hardness'.

Roughly speaking, all common asymmetric cryptosystems are based on one of the following two problems, or a variation thereof:

Integer factorization. Given a composite integer $n > 0$, find integers $u > 1$ and $v > 1$ such that $n = uv$.

In factoring based asymmetric cryptosystems, a user's public key contains the integer n and the corresponding private key contains (information equivalent to) u and v. The integer n is unique per user.

Discrete logarithm. Given an element g of a multiplicative written group G and an element h in the subgroup $\langle g \rangle$ generated by g, find an integer k such that $g^k = h$. The smallest non-negative such k is referred to as the discrete logarithm of h with respect to g and denoted $\log_g h$.

For additively written groups one would look for an integer k such that $kg = h$. The smallest non-negative such k is again referred to as $\log_g h$.

In discrete logarithm based asymmetric cryptosystems, a user's public key contains g and h and the corresponding private key contains $\log_g h$. Different users may share the same g but use different h's.

The *traditional* discrete logarithm problem refers to the case where G is chosen as the multiplicative group $(\mathbf{F}_{p^\ell})^*$ of a finite field $\mathbf{F}_{p^\ell}$ of cardinality p^ℓ, for some prime p and positive integer ℓ.

Instances of these problems can easily be generated that are suitable for cryptographic applications and generally believed to be hard to solve. In the sections below it is discussed how to do this in such a way that the corresponding cryptosystems offer adequate protection until a specified year, as defined in Section 2. This has certain consequences for the size of the integer n and its factors, for the cardinality $\#\langle g\rangle$ of the subgroup $\langle g\rangle$, and for the cardinality $p^\ell - 1$ of the group $G = (\mathbf{F}_{p^\ell})^*$ if the traditional discrete logarithm problem is used. Intuitively this is rather obvious, since small integers are easy to factor, small factors are easy to find, and discrete logarithms are easy to calculate if $\#\langle g\rangle$ is small. In Section 6 the requirements on n and its factors are discussed, and in Section 7 the same is done for g both for the case $G = (\mathbf{F}_{p^\ell})^*$ and for more general groups G.

Other asymmetric cryptosystems. There are quite a few asymmetric systems that are based on different mathematical problems than the currently popular ones mentioned above, but that have not yet gained general acceptance. The reason for the latter is usually related to the underlying mathematical problem, the cryptographic protocol it is embedded in, or a combination of these issues. There may be skepticism about the difficulty of the mathematical problem because it has not been studied long enough. Or the effectiveness of solution methods may be hard to judge or in a constant state of flux, making it difficult to recommend secure parameter choices. Also, cryptographic protocols that are provably equivalent to the mathematical problem may still be lacking, or the system may simply be too impractical. Asymmetric systems that have any of these shortcomings are not further discussed in this chapter. The reader is recommended to consult the recent cryptology literature to find the latest updates on asymmetric systems that are not treated here.

6 Factoring based systems

There are several types of asymmetric cryptosystems that rely for their security on the hardness of the integer factorization problem: if the integer factorization problem can be solved for a certain composite integer referred to as the *modulus* n, then the cryptosystem using that n can be broken. Thus, factoring the modulus suffices to break the system. In this section its is discussed how n should be selected in such a way that the integer factorization problem for n offers adequate protection until a year

of one's choice. It should be kept in mind, however, that for most common factoring based cryptosystems (such as RSA) it has, in general, not been proved that factoring the modulus is also necessary to break them, although such systems do exist.

Main variants. The way the modulus is constructed depends on the factoring based cryptosystem one uses. In the most common factoring based cryptosystems the modulus is the product of two primes of approximately the same size (cf. [39]). A variation, RSA multiprime (cf. [39]), improves the efficiency of the private operations by allowing more than two factors of approximately equal size in the modulus. Less common variants are RSA for paranoids [42], where the private operations are performed modulo the smallest prime factor of the modulus, and variants where the modulus contains repeated factors. Requirements on the size of the modulus and its factors are discussed below. For any of the variants moduli can be constructed efficiently because primes of any practical size can be generated quickly.

Trial division. The conceptually most straightforward way to factor a composite integer n is by trying if n is divisible by $2, 3, 5, 7, 11, 13, \ldots$, successively trying all primes until the smallest proper divisor is found. This process is known as *trial division*. It remains the method of choice of amateur-cryptanalysts. For that reason a detailed explanation of the cryptanalytic ineffectiveness of trial division is provided.

For random composites without known properties, i.e., composites not stemming from cryptographic applications, trial division is on average over the inputs the most efficient factoring algorithm because random composites can be expected to have a small factor: half of the random composities are even, so the first trial division attempt will be successful in 50% of the cases, one third of the remaining (odd) numbers is divisible by three, etc. It is very easy, however, to construct composites for which trial division is totally ineffective. This can be seen as follows.

According to the *prime number theorem* the number of primes up to x is proportional to $\frac{x}{\log x}$. This means that, to find the smallest prime factor p of n using trial division, on the order of $\frac{p}{\log p}$ smaller primes have to be tested before p is found. Because the cost of each attempt is at least proportional to the logarithm of the number tested, the overall computational effort to find p is proportional to p itself. Thus, if n is constructed as the product of two, say, b-digit primes, the computational effort to factor n using trial division is on the order of 10^b. Even for moderate b such as 50 a computational effort of this magnitude is out of

reach. Furthermore, there are other factoring methods that would factor such n much faster.

Another consequence of the prime number theorem is that the number of b-digit primes outnumbers the number of smaller primes. Thus, it does not help much, as often proposed, to exclude from the search in the example the primes having fewer than b digits thereby limiting the trial divisions to b-digit primes.

Exponential-time factoring algorithms. In the worst case where n has two factors of approximately equal size the computational effort to factor n using trial division is proportional to $\sqrt{n} = n^{1/2} = \exp((\log n)/2)$. With a constant multiple of the input length $\log_2 n$ in the exponent, it follows that trial division is an *exponential-time algorithm*. There are exponential-time factoring algorithms that are much faster than trial division. For instance, Pollard's rho method [35] can be expected to find the smallest p dividing n after a computational effort that is not proportional to p but to $\sqrt{p}$, i.e., proportional to $n^{1/4}$ in the worst case $p \approx \sqrt{n}$.

If exponential-time algorithms were the fastest factoring algorithms, it would be possible to select moduli n in such a way that $\log_2 n$ is proportional to the desired security level: if Pollard-rho would be the best factoring algorithm, then 4λ-bit moduli would offer security level λ. Unfortunately for cryptographic applications of factoring based asymmetric cryptosystems, exponential-time algorithms are by no means the best that can be done for factoring. As indicated above, much faster factoring algorithms exist. As a consequence, the required modulus bit length grows much faster than a linear function of the desired security level. In particular, modulus sizes grow much faster than symmetric cryptosystem key sizes and cryptographic hash function sizes.

Polynomial-time factoring algorithms. On the opposite side of the spectrum from exponential-time algorithms are *polynomial-time algorithms*: a polynomial-time factoring algorithm would require computational effort proportional to at most $(\log n)^c$, for some constant c. Although a polynomial-time factoring algorithm has been published in [45], it requires a not-yet-existing type of computer, a so-called *quantum computer*, to run it on. If the engineering problems of building a large enough quantum computer can be solved, factoring may be done in polynomial time, which will most likely mean the end for factoring based asymmetric cryptosystems. Alternatively, development of a polynomial-time factoring algorithm that would run on a traditional computer, a possibility that cannot yet provably be excluded, would have the same consequence. At this point there is not sufficient reason to suspect that practical polynomial-

time factoring is a realistic prospect. The possibility of practical polynomial time factoring is therefore not included in the analysis below.

What can realistically be done, however, is something that lies between exponential-time and polynomial-time factoring. These so-called subexponential-time factoring algorithms are further discussed below.

Subexponential-time factoring algorithms. The computational effort required for an exponential-time factoring algorithm is bounded from above by a constant positive power of

$$n = \exp(\log n).$$

For a polynomial-time method the required computational effort would be bounded from above by a constant power of

$$\log n = \exp(\log \log n).$$

To express the computational effort of algorithms that are faster than exponential time but not as fast as polynomial time, both possibilities are captured in a single formula in the following way. Let

$$L[n, r, \alpha] = \exp(\alpha(\log n)^r (\log \log n)^{1-r}).$$

Exponential time is characterized by $r = 1$, polynomial time by $r = 0$, and everything in between, i.e., $0 < r < 1$ is referred to as *subexponential time* (with, in all cases, α a positive constant).

There are many factoring algorithms for which the computational effort is expected to be $L[n, 1/2, 1+o(1)]$ for $n \to \infty$ (i.e., asymptotically for n to infinity, the value of α approaches 1). For most of these algorithms the analysis is based on heuristic arguments, for some it can rigorously be proved. Note that, on the scale from $r = 0$ to $r = 1$ suggested above, $L[n, 1/2, 1 + o(1)]$, i.e., $r = 1/2$, is exactly halfway between exponential time and polynomial time. One example is the quadratic sieve factoring algorithm (QS) which can heuristically be expected to factor n, irrespective of any properties its factors may have, for a computational effort that behaves as $L[n, 1/2, 1 + o(1)]$ for $n \to \infty$ (cf. [37]). Another example is the elliptic curve method (ECM) which can heuristically be expected to find a factor p of n for a computational effort $(\log n)^2 L[p, 1/2, \sqrt{2} + o(1)]$ (cf. [29]); in the worst case $p \approx \sqrt{n}$ this becomes $L[n, 1/2, 1 + o(1)]$.

Number Field Sieve. Because so many quite different methods all share essentially the same expected computational effort $L[n, 1/2, 1+o(1)]$, this was suspected by some to be the 'ultimate' complexity of factoring. In

1988 these cryptographic dreams were shattered by John Pollard's invention of a new factoring algorithm, cf. Pollard's first article in [23]. The original version, now referred to as the *Special Number Field Sieve* (SNFS), was intended to factor the ninth Fermat number $F_9 = 2^{2^9} + 1$, a number that was indeed completely factored in 1990 [24]. The SNFS can be applied to numbers that allow a particularly 'nice' polynomial representation, such as F_9. Based on heuristic arguments the expected computational effort is $L[n, 1/3, 1.526 + o(1)]$. The generalized version, now referred to as the *Number Field Sieve* (NFS), factors any number n for a (heuristic) expected computational effort $L[n, 1/3, 1.923 + o(1)]$ (cf. [23]), which was later improved to $L[n, 1/3, 1.902 + o(1)]$ (cf. [9]).

On the scale from exponential time ($r = 1$) to polynomial time ($r = 0$) the NFS represents substantial progress from the halfway point ($r = 1/2$) in the direction of polynomial-time algorithms. Since the invention of the NFS no progress affecting the current best $r = 1/3$ value has been published (with the exception of $r = 0$ for quantum computers).

The cost of the NFS. Let the cost function be as defined in Section 2, i.e., the product of time (or, equivalently, computational effort) and equipment cost. The NFS has two major stages, the relation collection stage and the matrix stage. As shown in [2], the cost of the NFS depends on the way the relation collection stage is carried out. If a memory-intensive approach based on sieving is used the overall NFS cost behaves as $L[n, 1/3, 2.852 + o(1)]$ for $n \to \infty$. An ECM-based approach is asymptotically considerably less costly: just $L[n, 1/3, 1.976 + o(1)]$ for $n \to \infty$.

NFS results. Compared to the older $L[n, 1/2, 1 + o(1)]$-methods, the NFS is conceptually complicated and, originally, suffered from rather large $o(1)$-values. Therefore, it was believed by some that the NFS had only theoretical but no practical value. However, a lot of progress has been made to improve the method, thereby lowering the $o(1)$'s. As a result the NFS eventually surpassed the older methods also from a practical point of view. At the time of writing of this chapter, the NFS is the method of choice for actual large-scale factorization experiments (cf. [7] and [13]) and special purpose factoring hardware design proposals (cf. [44], [25], and [43]). The following results have been obtained using the sieving-based approach:

- **Software implementation:** a 576-bit modulus has been factored using the NFS in about 12 years of computing time on a 1GHz Pentium III processor [13] (in reality it was done on m such processors in $12/m$ years of computing time per processor, for some large m).

– **Special purpose hardware design proposal:** using 90 nanometer VLSI technology, it can be expected that factorization of a 1024-bit modulus takes at most one year using a special purpose hardware device that takes at most US$1 million to build [27].
It follows that in 2004, at the time of writing this chapter, the cost of factoring 1024-bit moduli can be estimated as at most 400M dollar-days.

Actually, these results refer to just the relation collection step, in practice the most cumbersome stage of the NFS factoring process. The other major stage, the matrix step, although in theory equally costly, is in practice negligible compared to the relation collection stage (cf. [2] and [26]).
Extrapolation to other modulus lengths. The 400M dollardays cost to factor 1024-bit moduli in the year 2004 is combined with the asymptotic cost estimates for NFS to estimate the cost of factoring b-bit moduli in 2004 as

$$\frac{L[2^b, 1/3, \alpha]}{L[2^{1024}, 1/3, \alpha]} \cdot 400\text{M dollardays},$$

with $\alpha = 2.852 + o(1)$. For the sake of simplicity—and because no better alternative is available—it is assumed that upon substitution the two $o(1)$'s cancel. From a theoretical point of view this assumption is hardly acceptable, but for limited range approximations the results of this compromise approach have been satisfactory, so far. Although the 400M dollardays for 1024-bit moduli is based on the sieving-based approach, rough estimates for the ECM-based approach are not that much different. Therefore one may alternatively replace 2.852 by 1.976 in the above estimate. For key length estimate purposes $\alpha = 1.976 + o(1)$ is a more prudent choice than $\alpha = 2.852 + o(1)$, because $\alpha = 1.976 + o(1)$ results in lower factoring costs and therefore larger and more conservative choices for key lengths achieving adequate protection.

As an example, 1248-bit moduli are roughly expected to be between

$$\frac{L[2^{1248}, 1/3, 1.976]}{L[2^{1024}, 1/3, 1.976]} \approx 250$$

and

$$\frac{L[2^{1248}, 1/3, 2.852]}{L[2^{1024}, 1/3, 2.852]} \approx 3000$$

times costlier to factor than 1024-bit ones. Similarly, 1536-bit moduli are between 137K and 26M times costlier and 2048-bit moduli are at least 2 billion times costlier to factor than 1024-bit ones.

Cryptanalytic developments. During the last three to four decades there has been a steady stream of developments in integer factorization algorithms. The practical performance of the best existing algorithms such as the NFS and the ECM is still constantly fine-tuned and improved. This smooth progress is, less frequently, combined with more substantial advances such as, most importantly, the invention of an entirely new method or, less dramatic but often with important practical consequences, better ways to handle certain steps of existing methods. It is reasonable to assume that the trend as observed so far will continue for the years to come.

Combining the occasional jumps and the regular smooth progress, the effect of cryptanalytic progress on the difficulty of the integer factorization problem turns out to be very similar to Moore's law: overall, and on the same equipment, the cost of factoring drops by a factor 2 every 18 months. According to Moore's traditional law as formulated in Section 2, the equipment cost also drops by a factor 2 every 18 months. These two effects, cryptanalytic progress and hardware advances, have in the past been independent and it is reasonable to assume that they will remain to be so. As a result of the combination of these two independent effects, the decrease in the cost of factoring is modelled in the following way:

Double Moore factoring law. The cost of factoring any fixed modulus drops by a factor 2 every 9 months.

As an example, in 2.5 years it can be expected that the cost of factoring a 1024-bit modulus is reduced to

$$\frac{400\text{M}}{2^{2.5\cdot 12/9}} \approx 40\text{M dollardays.}$$

Similarly, over a period of 6 years it is expected that the factoring cost drops by a factor $2^{6\cdot 12/9} = 256$. Thus, it would be conservative to expect that factoring a 1248-bit in 2010 would cost about the same as a 1024-bit modulus in 2004.

Small factors. In regular RSA the modulus is chosen as the product of two primes of approximately equal sizes. Asymptotically, and for all regular RSA moduli commonly in use, the most efficient published method to factor such moduli is the NFS. As cited above, there are at least two variants of RSA where the modulus n may have one (RSA for paranoids) or more (RSA multiprime) prime factors that are substantially smaller than $\sqrt{n}$. Currently the asymptotically fastest method to find small factors, if there are any, is the ECM. Therefore, care must be taken to select

the factors in such a way that finding them using the ECM can be expected to be at least as hard as factoring n using the NFS. The reader is referred to [22] for a further discussion of this point.

RSA modulus lengths that offer adequate protection. According to the definition in Section 2 an RSA modulus offers adequate protection until year y if the factorization cost in that year can be expected to be at least 40M dollardays. Thus, 1024-bit RSA moduli offer adequate protection for 2.5 more years from the year 2004, when this chapter was written. More in general, by combining the above extrapolation to other modulus lengths with the double Moore factoring law it can be determined—to the best of the current knowledge—if a b-bit RSA modulus offers adequate protection until the year y: it does if

$$\frac{L(2^b, 1/3, \alpha)}{L(2^{1024}, 1/3, \alpha)} \cdot 400 \geq 40 \cdot 2^{4(y-2004)/3},$$

where, again, $\alpha = 1.976$ leads to a conservative, relatively large b-value and $\alpha = 2.852$ to a less prudent smaller one.

Table 3 lists the RSA modulus bit-lengths for both choices for α and for several years, and Table 4 lists the years until which several common RSA modulus bit-lengths offer adequate protection, again for both α-values. For each year y in the two tables the security level $\lambda(y) = 56 + \frac{2(y-1982)}{3}$ that offers adequate protection until year y, rounded upwards to the nearest integer, is given between parentheses (cf. equation (2) in Section 2). Note that $\lambda(y)$ corresponds to the minimally required symmetric key length in year y.

Table 3. Minimal RSA modulus bit-lengths for protection until a given year.

year y ($\lambda(y)$)	(optimistic) bit-length for $\alpha = 2.852$	(conservative) bit-length for $\alpha = 1.976$
2010 (75)	1112	1153
2020 (82)	1387	1569
2030 (88)	1698	2064
2040 (95)	2048	2645
2050 (102)	2439	3314

It follows from the Tables that 2048-bit RSA moduli offer adequate protection at least until the year 2030, and even until 2040 if one is less prudent and confident that ECM-based factoring devices will not be able to outperform the sieving-based approach before the year 2040.

Although this type of estimates is the best that can be done at this point, it should be understood that actual factoring capabilities may fol-

Table 4. Common RSA modulus bit-length life spans.

modulus bit-length	(conservative) year y_c for $\alpha = 1.976$ ($\lambda(y_c)$)	(optimistic) year y_o for $\alpha = 2.852$ ($\lambda(y_o)$)
1024	2006 (72)	2006 (72)
1280	2014 (78)	2017 (80)
1536	2020 (82)	2025 (85)
2048	2030 (88)	2040 (95)
3072	2046 (99)	2065 (112)
4096	2060 (108)	2085 (125)
8192	2100 (135)	2142 (163)

low an entirely different pattern. Any prediction more than a few decades away about security levels is wishful thinking. The figures in the tables should be properly interpreted, namely as today's best estimates that may have to be revised tomorrow. Anyone using factoring based asymmetric cryptosystems should constantly monitor and stay ahead of the developments in the research community.

7 Discrete logarithm based systems

Let g belong to some group G and let h be an arbitrary element of the subgroup $\langle g \rangle$ of G generated by g of known order $\#\langle g \rangle$. The cryptographic application of the generator g imposes a unique representation for each element of G and thus of $\langle g \rangle$. Given this representation the group operation and inversion can be performed efficiently. Using multiplicative notation for the group operation this implies that for any k the element g^k can be computed in $O(\log k)$ group operations and at most a single inversion. On the other hand, because of the cryptographic application, g and G must be chosen such that the 'reverse' problem of computing $\log_g h$ offers adequate protection until a year of one's choice. The resulting requirements on g and G are discussed in this section.

The discrete logarithm problem can be solved either in the subgroup $\langle g \rangle$ directly or in the group G in which $\langle g \rangle$ is embedded. For adequate protection it must be infeasible to solve the problem using either approach. Of particular practical interest is the traditional discrete logarithm problem where $G = (\mathbf{F}_{p^\ell})^*$.

Unsuitable groups. There are groups in which discrete logarithms are not hard to compute. An example is the additive group of integers modulo a positive integer, where computing discrete logarithms is equivalent to modular division. Obviously, such groups must be avoided in cryptographic applications. Unfortunately, this is not always as easy as it sounds. There are examples of groups where at first sight the discrete

logarithm problem looks hard, but where, after closer scrutiny by the research community, the problem turned out to be easier than expected. For instance, a certain type of elliptic curve based groups as proposed for cryptographic applications in [31] was shown to allow trivial discrete logarithm computation in [40], [41] and [47]. Interestingly, these groups were offered as an alternative to another class of elliptic curve based groups where the discrete logarithm problem allowed an undesirable reduction to the traditional case $G = (\mathbf{F}_{p^\ell})^*$ (cf. [14] and [30]).

Accidents of this sort are impossible to avoid. But, as a general advice, cryptographic application of newly proposed groups should be postponed until the mathematical and cryptanalytic communities have scrutinized the proposed groups and failed to 'break' them. In the sequel it is implicitly assumed that the groups in question do not allow other attacks than the generic ones described below.

The discrete logarithm problem in $G = (\mathbf{F}_{p^\ell})^*$. If $G = (\mathbf{F}_{p^\ell})^*$, discrete logarithms in G can be calculated using a method that is similar to the NFS algorithm for integer factorization discussed in Section 6. Roughly speaking, computing discrete logarithms in $(\mathbf{F}_{p^\ell})^*$ is about as hard as factoring an integer n with $\log n \approx \log p^\ell$ using the NFS. Thus, to achieve adequate protection until a given year the size requirements on n as presented in Section 6 imply the same size requirements on p^ℓ.

This is a rough estimate in the sense that it somewhat underestimates the difficulty of computing discrete logarithms in $(\mathbf{F}_{p^\ell})^*$ and thereby overestimates the p^ℓ values that would suffice for adequate protection. An often encountered argument is that the matrix step (cf. Section 6) as required for the discrete logarithm version of the NFS, is much harder than the one for the regular factoring NFS. It is true that the matrices, assuming comparable dimensions, are harder to deal with. But, in the first place, compared to factoring the cost will not increase by more than a factor $(\log \#\langle g\rangle)^2$, which is, relatively speaking, only a minor effect. In the second place, the actual cost of the relation collection stage (cf. Section 6) may still far outweigh the matrix step cost, further diminishing the effect of the more expensive matrix step on the overall cost of the computation. Given the granularity of finite field sizes that are available in practice, there is no practical need for more accurate estimates.

Reduction to prime order subgroup. Due to the Pohlig-Hellman algorithm, the problem of computing $\log_g h$ can efficiently be reduced to the problem of computing $\log_g h$ modulo each of the prime divisors of $\#\langle g\rangle$ (cf. [36]). Therefore, and because the complete factorization of $\#\langle g\rangle$ may be unknown and hard to find (cf. Section 6), it is assumed that $\#\langle g\rangle$ has

at least one prime divisor that satisfies the further requirements specified below. For convenience of presentation and without loss of generality, it is assumed that $\#\langle g\rangle$ itself is prime. If $G = (\mathbf{F}_{p^\ell})^*$ with $\ell \geq 2$ this prime $\#\langle g\rangle$ must be carefully chosen, as shown in the next paragraphs.

The discrete logarithm problem in a subgroup of $G = (\mathbf{F}_{p^\ell})^*$. The generator g belongs to $G = (\mathbf{F}_{p^\ell})^*$ and thus has (prime) order dividing $p^\ell - 1$. For $\ell > 1$, however, the number $p^\ell - 1$ has factors that should be avoided in the sense that if $\#\langle g\rangle$ divides such a factor, the difficulty of the discrete logarithm problem in $\langle g\rangle$ is affected. This is explained below.

For each positive integer d dividing ℓ the finite field $\mathbf{F}_{p^\ell}$ has a subfield $\mathbf{F}_{p^d}$ and the multiplicative group $G = (\mathbf{F}_{p^\ell})^*$ has a subgroup $(\mathbf{F}_{p^d})^*$ of order $p^d - 1$ dividing $p^\ell - 1$. If the order $\#\langle g\rangle$ of g divides $p^d - 1$ for a d less than and dividing ℓ, then g belongs to the true subgroup $(\mathbf{F}_{p^d})^*$ of the multiplicative group $G = (\mathbf{F}_{p^\ell})^*$ of the finite field $\mathbf{F}_{p^\ell}$, and thereby g belongs to the true subfield $\mathbf{F}_{p^d}$ of $\mathbf{F}_{p^\ell}$. Representations of such subfield elements of $\mathbf{F}_{p^\ell}$ can efficiently be mapped back and forth to direct representations in the finite field $\mathbf{F}_{p^d}$ itself. As a result, the discrete logarithm problem in $\langle g\rangle$ can be solved in the true subfield $\mathbf{F}_{p^d}$, which is a substantially easier problem than in the 'large' field $\mathbf{F}_{p^\ell}$: in the notation of Section 6 it reduces the cost of computing discrete logarithms from $L[p^\ell, 1/3, \alpha]$ to $L[p^d, 1/3, \alpha]$, for some constant $\alpha > 0$.

It follows that g should be chosen in such a way that its order $\#\langle g\rangle$ does not divide $p^d - 1$ for any d less than and dividing ℓ. This is achieved as follows. The dth cyclotomic polynomial $\Phi_d(X)$ is recursively defined by

$$X^d - 1 = \prod_{t \text{ dividing } d} \Phi_t(X).$$

For instance, $\Phi_1(X) = X - 1$, $\Phi_2(X) = \frac{X^2-1}{X-1} = X + 1$, $\Phi_3(X) = \frac{X^3-1}{X-1} = X^2 + X + 1$, etc. Thus, g must be chosen in such a way that $\#\langle g\rangle$ divides $p^\ell - 1$ but does not divide $\Phi_d(p)$ for a d less than and dividing ℓ. This condition is satisfied if g is chosen so that $\#\langle g\rangle$ is a prime divisor larger than ℓ of $\Phi_\ell(p)$, the 'last' cyclotomic factor of $p^d - 1$ (cf. [21]). For instance, if $\ell = 2$ the order $\#\langle g\rangle$ of g must be chosen as a sufficiently large prime divisor of $\Phi_2(p) = p + 1$; and of

$$\Phi_6(p) = \frac{p^6 - 1}{(p-1)(p+1)(p^2+p+1)} = p^2 - p - 1$$

if $\ell = 6$.

Size requirements. Under the general representation assumptions specified at the beginning of this section (and avoiding unsuitable groups), the

best methods to solve the discrete logarithm problem in $\langle g \rangle$ require approximately $\sqrt{\#\langle g \rangle}$ group operations. There are essentially two methods that achieve this operation count, Shanks' baby-step-giant-step method [18, Exercise 5.17] and Pollard's rho method [35]. Shanks' method requires a substantial amount of memory. This implies that the cost of an attack effort (as defined in Section 2) by means of Shanks' methods is much larger than $\sqrt{\#\langle g \rangle}$: according to [49] it is approximately $(\#\langle g \rangle)^{2/3}$.

Pollard's rho method, on the other hand, requires just a constant amount of memory when run on a single processor. Although this implies an attack effort cost of approximately $\sqrt{\#\langle g \rangle}$, an attack of this sort does not have any practical significance. However, a variation of Pollard's rho method allows efficient parallelization with the same cost (cf. [49]). Therefore, both from a theoretical as practical point of view, the cost of Pollard-rho based attack effort is approximately $\sqrt{\#\langle g \rangle}$.

It follows that the discrete logarithm problem in an order $\#\langle g \rangle$ subgroup g offers security level approximately $\log_2 \sqrt{\#\langle g \rangle} = \log_4 \#\langle g \rangle$. To decide if a certain discrete logarithm security level offers adequate protection as defined in Section 2, the relative speed of the group operation compared to the DES must in principle be taken into account. Since in any standard application the DES will be at least as fast as the group operation, and considerably faster if $g \in (\mathbf{F}_{p^\ell})^*$, neglecting this effect will only increase the level of protection offered by the discrete logarithm based system.

Cryptanalytic developments. Concerning cryptanalytic methods that directly attack the subgroup discrete logarithm problem, the most recent substantial cryptanalytic development was the parallelization of Pollard's rho method, as referred to above. This influenced the practical significance of a Pollard-rho based attack, but had no theoretical effect on the cost. As far as the choice of the subgroup size $\#\langle g \rangle$ is concerned, it is therefore reasonable to assume that for the foreseeable future the cost of subgroup attack efforts will not be different from the current cost of $\sqrt{\#\langle g \rangle}$ group operations. This cost corresponds to the provable lower bound for the computation of discrete logarithms in generic groups (cf. [34] and [46]).

If special properties of $g \in (\mathbf{F}_{p^\ell})^*$ are taken into account, there has been a steady stream of improvements to the NFS method for factoring that may have comparable effects on the version of the NFS that applies to the computation of discrete logarithms in $(\mathbf{F}_{p^\ell})^*$. As in Section 6 it is reasonable to assume that in the foreseeable future there will not be major variations in the rate of cryptanalytic progress observed over the last few decades.

Choices of $\#\langle g\rangle$ and p^ℓ that offer adequate protection. Summarizing the above conditions on g, it is assumed that g is chosen in such a way that $\#\langle g\rangle$ is prime, so that the discrete logarithm problem in $\langle g\rangle$ cannot be reduced to a discrete logarithm problem in a smaller group (of order a proper divisor of $\#\langle g\rangle$). Furthermore, if $g \in (\mathbf{F}_{p^\ell})^*$ it is assumed that $\#\langle g\rangle$ is a prime divisor larger than ℓ of $\Phi_\ell(p)$ to make sure that g cannot be embedded in a smaller multiplicative group $(\mathbf{F}_{p^d})^*$ for some $d < \ell$.

Under these restrictions, g must be chosen such that the discrete logarithm problem in $\langle g\rangle$ offers adequate protection until the year of one's choice. Combining the attack effort cost of $\sqrt{\#\langle g\rangle}$ with Moore's law it follows that a subgroup of prime order $\#\langle g\rangle$ offers adequate protection until the year

$$y(\log_4 \#\langle g\rangle) = 1982 + \frac{3(\log_4 \#\langle g\rangle - 56)}{2}$$

(cf. equation (1) in Section 2). This 'double growth' compared to symmetric key lengths leads to the same rule of thumb as given at the end of Section 4 for hash function lengths. Since the collision attack in Section 4 and the Pollard-rho based attack here are both based on the same 'birthday paradox' technique, this is hardly a surprise.

If $g \in (\mathbf{F}_{p^\ell})^*$ adequate protection until year y also requires to select p^ℓ in such a way that

$$\frac{L(p^\ell, 1/3, \alpha)}{L(2^{1024}, 1/3, \alpha)} \cdot 400 \geq 40 \cdot 2^{4(y-2004)/3},$$

with α either 1.976 (prudent) or 2.852 (optimistic) as in Section 6. This is the same requirement as on regular RSA moduli (cf. Section 6).

It follows that the US Government's Digital Signature Algorithm (DSA) with $\#\langle g\rangle \approx 2^{160}$ offers adequate protection against subgroup attacks until the year

$$y(\log_4 \#\langle g\rangle) \approx y(\log_4 2^{160}) = y(80) = 2018.$$

But the fact that the DSA prescribes usage of $g \in (\mathbf{F}_p)^*$ with $\log_2 p \leq 1024$ undermines the security level and implies that DSA offers adequate protection only until 2006 (cf. Table 4 in Section 6). ECDSA (cf. [16]), on the other hand, does not suffer from an embedding in a finite field and is believed to offer adequate protection until 2018 when 160-bit prime order subgroups are used.

8 Conclusion

To summarize, adequate protection was defined as the security offered in 1982 by the DES. It was argued that a system offers adequate protection until a given year if the cost of a successful attack in that year is at least 40M dollardays: a computation that lasts x days on a piece of equipment that costs $40/x$ million dollars to build (for any reasonable x).

Given this definition, for the most common cryptographic systems the following general key length recommendations can be made.

Symmetric cryptosystems. A symmetric cryptosystem with $(56 + b)$-bit keys and no known weaknesses offers adequate security until year $1982 + y$ only if $3b \geq 2y$.

Cryptographic hash functions. A cryptographic hash function of bit-length $112 + b$ and without known weaknesses offers adequate security until year $1982 + y$ only if $3b \geq 4y$.

Factoring based asymmetric cryptosystems. Refer to Tables 3 for modulus bit-lengths that should offer adequate protection until year $2000+10i$ for $0 < i \leq 5$. Refer to Table 4 for the year until which several common modulus bit-lengths can be expected to offer adequate protection.

Discrete logarithm based asymmetric cryptosystems. A subgroup $\langle g \rangle$ offers adequate security until year $1982 + y$ only if

$$3(\log_4(\#\langle g \rangle) - 56) \geq 2y.$$

If $g \in (\mathbf{F}_{p^\ell})^*$, then $\log_2 p^\ell$ must satisfy the same requirements as modulus bit-lengths for factoring based asymmetric cryptosystems. Furthermore, stay away from newly proposed groups.

Finally, it was shown how the definition of adequate protection can be tuned to one's own perception of security and how this changes the key length recommendations.

References

1. M. Bellare, P. Rogaway, *Collision-resistant hashing: towards making UOWHFs practical*, Proceedings Crypto'97, LNCS 1294, Springer-Verlag 1997, 470–484.
2. D.J. Bernstein, *Circuits for integer factorization: a proposal*, manuscript, November 2001; available at `cr.yp.to/papers.html#nfscircuit`.
3. E. Biham, O. Dunkelman, V. Furman, T. Mor, *Preliminary report on the NESSIE submissions Anubis, Camelia, IDEA, Khazad, Misty1, Nimbus, Q*, available from https://www/cosic.esat.kuleuven.ac.be/nessie/reports.
4. A. Biryukov, A. Shamir, D. Wagner, *Real time cryptanalysis of A5/1 on a PC*, Proceedings of FSE 2000, LNCS 1978, Springer-Verlag 2001, 1–18.

5. 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.
6. J.R.T. Brazier, *Possible NSA decryption capabilities*, jya.com/nsa-study.htm.
7. 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, Springer-Verlag 2000, 1–17.
8. F. Chabaud, A. Joux, *Differential collisions in SHA-0*, Proceedings Crypto'98, LNCS 1462, Springer-Verlag 1998, 56–71.
9. D. Coppersmith, *Modifications to the number field sieve*, J. Crypto. **6** (1993) 169–180.
10. H. Dobbertin, A. Bosselaers, B. Preneel, *RIPEMD-160, a strengthened version of RIPEMD*, Fast Software Encryption, LNCS 1039, Springer-Verlag 1996, 71–82.
11. Electronic Frontier Foundation, *Cracking DES*, O'Reilly, San Francisco, July 1998.
12. S. Fluhrer, I. Mantin, A. Shamir, *Attacks on RC4 and WEP*, RSA Laboratories' Cryptobytes, v. 5, no 2 (2001) 26–34; also at www.rsasecurity.com/rsalabs/cryptobytes.
13. J. Franke, personal communication, January 2004.
14. G. Frey, H.-G. Rück, *A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves*, Math. Comp. **62** (1994) 865–874.
15. H. Handschuh, H. Gilbert, x^2 *Cryptanalysis of the SEAL encryption algorithm*, Proceedings of FSE 1997, LNCS 1267, Springer-Verlag 1997, 1–12.
16. D. Johnson, A. Menezes, *The elliptic curve digital signature algorithm (ECDSA)*, CACR Technical report CORR 99-31, University of Waterloo, 1999.
17. J. Kilian, P. Rogaway, *How to protect DES against exhaustive key search*, Proceedings Crypto'96, LNCS 1109, Springer-Verlag 1996, 252–267.
18. D.E. Knuth, *The art of computer programming, Volume 2, Seminumerical Algorithms*, third edition, Addison-Wesley, 1998.
19. P.C. Kocher, *Breaking DES*, RSA Laboratories' Cryptobytes, v. 4, no 2 (1999) 1–5; also at www.rsasecurity.com/rsalabs/cryptobytes.
20. S. Landau, *Polynomials in the nation's service: using algebra to design the advanced encryption standard*, The mathematical society of America monthly, **111** (2004) 89–117.
21. A.K. Lenstra, *Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields*, Proceedings ACISP'97, LNCS 1270, Springer-Verlag 1997, 127–138.
22. A.K. Lenstra, *Unbelievable security*, Proceedings Asiacrypt 2001, LNCS 2248, Springer-Verlag 2001, 67–86.
23. A.K. Lenstra, H.W. Lenstra, Jr., (eds.), *The development of the number field sieve*, Lecture Notes in Math. **1554**, Springer-Verlag 1993.
24. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, J.M. Pollard, *The factorization of the ninth Fermat number*, Math. Comp. **61** (1993) 319–349.
25. A.K. Lenstra, A. Shamir, *Analysis and optimization of the TWINKLE factoring device*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 35–52.
26. A.K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer, *Analysis of Bernstein's factorization circuit*, Proceedings Asiacrypt 2002, LNCS 2501, Springer-Verlag 2002, 1–26.
27. A.K. Lenstra, E. Tromer, A. Shamir, W. Kortsmit, B. Dodson, J. Hughes, P. Leyland, *Factoring estimates for a 1024-bit RSA modulus*, Proceedings Asiacrypt 2003, LNCS 2894, Springer-Verlag 2003, 55–74.

28. A.K. Lenstra, E.R.Verheul, *Selecting Cryptographic Key Sizes*, Proceedings PKC 2000, LNCS 1751, Springer-Verlag 2000, 446–465; J. Crypto. **14** (2001) 255–293; available from www.cryptosavvy.com.
29. H.W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. **126** (1987) 649–673.
30. A.J. Menezes, T. Okamoto, S.A. Vanstone, *Reducing elliptic curve logarithms to a finite field*, IEEE Trans. Info. Theory **39** (1993) 1639–1646.
31. A. Miyaji, *Elliptic curves over* $\mathbf{F}_p$ *suitable for cryptosystems*, Proceedings Auscrypt 92, LNCS 718, Springer-Verlag 1993, 479–491.
32. National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard." National Bureau of Standards, U.S. Department of Commerce, January 1997.
33. NESSIE, New European schemes for signatures, integrity, and encryption, 2000–2003, https: //www.cosic.esat.kuleuven.ac.be/nessie/.
34. V.I. Nechaev, *Complexity of a determinate algorithm for the discrete logarithm*, Mathematical Notes, **55** (1994) 155–172; translated from Matematicheskie Zametki, 55(2) (1994) 91–101; this result dates from 1968.
35. J.M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. Comp. **32** (1978) 918–924.
36. S.C. Pohlig, M.E. Hellman, *An improved algorithm for computing logarithms over* $GF(p)$ *and its cryptographic significance*, IEEE Trans. Info. Theory **24** (1978) 106–110.
37. C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, in *Computational methods in number theory* (H.W. Lenstra, Jr., R. Tijdeman, eds.) Math. Centre Tracts **154**, **155**, Mathematisch Centrum, Amsterdam (1983) 89–139.
38. http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html.
39. R.L. Rivest, A. Shamir, L.M. Adleman, *Cryptographic communications system and method*, U.S. Patent 4,405,829, 1983.
40. T. Satoh, K. Araki, *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, Comm. Math. Univ. Sancti Pauli, **47** (1998) 81–92.
41. I.A. Semaev, *Evaluation of discrete logarithms on some elliptic curves*, Math. Comp. **67** (1998) 353–356.
42. A. Shamir, *RSA for paranoids*, RSA Laboratories' Cryptobytes, v. 1, no. 3 (1995) 1–4.
43. A. Shamir, E. Tromer, *Factoring large numbers with the TWIRL device*, Proceedings Crypto 2003, LNCS 2729, Springer-Verlag 2003, 1–26.
44. A. Shamir, *Factoring large numbers with the TWINKLE device*, Proceedings CHES'99, LNCS 1717, Springer-Verlag, 1999.
45. 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.
46. V. Shoup, *Lower bounds for discrete logarithms and related problems*, Proceedings Eurocrypt'97, LNCS 1233,
47. N.P. Smart, *The discrete logarithm problem on elliptic curves of trace one*, J. Crypto. **12** (1999) 193–196.
48. M.J. Wiener, *Efficient DES key search*, manuscript, Bell-Northern Research, August 20, 1993.
49. M.J. Wiener, *The full cost of cryptanalytic attacks*, accepted for publication in J. Crypto.