Unbelievable Security
Matching AES security using public key systems

Arjen K. Lenstra

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

Abstract. The Advanced Encryption Standard (AES) provides three levels of security: 128, 192, and 256 bits. Given a desired level of security for the AES, this paper discusses matching public key sizes for RSA and the ElGamal family of protocols. For the latter both traditional multiplicative groups of finite fields and elliptic curve groups are considered. The practicality of the resulting systems is commented upon. Despite the conclusions, this paper should not be interpreted as an endorsement of any particular public key system in favor of any other.

1 Introduction

The forthcoming introduction [12] of AES-128, AES-192, and AES-256 creates an interesting new problem. In theory, AES-128 provides a very high level of security that is without doubt good enough for any type of commercial application. Levels of security higher than AES-128, and certainly those higher than AES-192, are beyond anything required by ordinary applications. Suppose, nevertheless, that one is not satisfied with the level of security provided by AES-128 and insists on using AES-192 or AES-256. This paper considers the question what key sizes of corresponding security one should then be using for the following public key cryptosystems:

- RSA and RSA multiprime (RSA-MP; the earliest reference is [14]).
- Diffie-Hellman and ElGamal-like systems [10, 15] based on the discrete logarithm problem in prime order subgroups of
  - multiplicative groups of prime fields.
  - multiplicative groups of extension fields: fields of fixed small characteristic and compressed representation methods (LUC [17] and XTR [8]).
  - groups of elliptic curves over prime fields (ECC, [1]).

These are the most popular systems and the only ones that are widely accepted. Systems that have recently been introduced and that are still under scrutiny are not included, with the exception of XTR – it is included because this paper sheds new light on its alleged performance equivalence to ECC. Also discussed are performance issues related to the usage of keys of the resulting sizes.

The introduction of the AES will soon bring along the introduction of cryptographic hash functions of matching security levels [13], namely SHA-256, SHA-384, and SHA-512. Because many common subgroup based cryptographic protocols use subgroup orders and hashes of the same sizes, the decision what subgroup size to use with AES-ℓ becomes easy: use subgroups of prime order q with $\lceil \log_2 q \rceil = 2\ell$. For ECC that settles the issue, from a practical point of view at least. This is reflected in the revised standard FIPS 186-2 [11]. For the other subgroup systems the finite field size remains to be decided upon. It may be assumed that both for properly chosen finite fields and for ECC the resulting subgroup operation is slower than a single application of the AES or SHA. It follows that, with respect to the familiar exhaustive search, collision, and square-root attacks against AES-ℓ, SHA-2ℓ, and properly chosen subgroups, respectively, the weakest links will be the AES and the SHA, not the subgroup based system.

It may be argued that the question addressed in this paper is of academic interest only. Indeed, it remains to be seen if the security obtained by actual realization and application of 'unbelievably secure' systems such as AES-192, AES-256, or matching public key systems, will live up to the intended theoretical bounds. That issue is beyond the scope of this article. Even under the far-fetched assumption that implementations are perfect, it is conceivable that the actual security achieved by the AES is less than the intended one. Thus, even though one may be happy with the (intended) security provided by AES-128, one may cautiously decide to use AES-256 and match it with a public key system of 'only' 128-bit security [21]. Therefore, and to give the theme of this paper somewhat wider applicability, not only public key sizes matching AES-192 and AES-256 are presented, but also the possibly more realistic sizes matching DES, 2K3DES, 3K3DES, and AES-128. Here iK3DES refers to triple DES with i keys.

This paper is organized as follows. Issues concerning security levels of the cryptosystems under consideration are discussed in Section 2. RSA moduli sizes of security equivalent to the symmetric systems, now and in the not too distant future, are presented in Section 3. The security of RSA-MP, i.e., the minimal factor size of (matching) RSA moduli, is discussed in Section 4. Section 5 discusses matching finite field sizes for a variety of finite fields as applied in systems based on subgroups of multiplicative groups (i.e., not ECC): prime fields, extension fields with constant extension degree, and fields with constant (small) characteristic. Section 6 discusses various performance related issues, such as total key lengths and relative runtimes of cryptographic operations. A summary of the findings is presented in Section 7.

2 Security levels

2.1 Breaking cryptosystems. Throughout this paper breaking a symmetric cryptosystem means retrieving the symmetric key. Breaking RSA means factoring the public modulus, and breaking a subgroup based public key system means computing the discrete logarithm of a public subgroup element with respect to a known generator. Attacks based on protocol specific properties or the size of

public or secret exponents are not considered. Thus, this paper lives in an idealized world where only key search and number theoretic attacks count. For any real life situation this is a gross oversimplification. But real life security cannot be obtained without resistance against these basic attacks.

2.2 Equivalence of security. Under the above attack model, two cryptosystems provide the same level of security if the expected effort to break either system is the same. This way of comparing security levels sounds simpler than it is, because 'effort' can be interpreted in several ways. In [7] two possible ways are distinguished to compare security levels:

- Two cryptosystems are *computationally equivalent* if breaking them takes, on average, the same computational effort.
- Two cryptosystems are *cost equivalent* if acquiring the hardware to break them in the same expected amount of time costs the same.

Both types of equivalence have their pros and cons. The computational effort to break a cryptosystem can, under certain assumptions, be estimated fairly accurately. If the assumptions are acceptable, then the outcome should be acceptable as well. Computational effort does not take into account that it may be possible to attack one systems using much simpler and cheaper hardware than required for the other. The notion of cost equivalence attempts to include this issue as well. But it is an inherently much less precise measure, because cost of hardware can impossibly be pinpointed.

2.3 Symmetric key security levels. A symmetric cryptosystem provides d-bit security if breaking it requires on average 2^{d-1} applications of the cryptosystem. Throughout this paper the following assumptions are made:

1. Single DES provides 56-bit security.
2. 2K3DES provides 95-bit security.
3. 3K3DES provides 112-bit security [15, page 360].
4. AES-ℓ provides ℓ-bit security, for $\ell = 128, 192, 256$.

The single DES estimate is based on the effort spent by recent successful attacks on single DES, such as described in [5]. The 2K3DES estimate is based on the approximately 100-bit security estimate from [20] combined with the observation that since 1990 the price of memory has come down relative to the price of processors. It may thus be regarded as an estimate that is good only for cost equivalence purposes. However, the computationally equivalent estimate may not be much different. The commonly used 112-bit estimate for 3K3DES is of a computational nature and ignores memory costs that far exceed processor costs. The best realistic attack uses parallel collision search on a machine with about a million terabytes of memory, and would lead to a security level of 116 bits[1]. This is more conservative than the classic meet-in-the-middle attack, which would lead to 128-bit (cost-equivalent) security. These comments on 2K3DES and 3K3DES security levels are due to Mike Wiener [21].

[1] Each 4-fold memory reduction doubles the runtime.

As far as the AES estimates are concerned, there is no a priori reason to exclude the possibility of substantial cryptanalytic progress affecting the security of the AES, in particular given how new the AES is. It is assumed, however, that if the AES estimates turn out to be wrong, then the AES will either be patched (cf. the replacement of SHA by SHA-1), or that it will be replaced by a new version of the proper and intended security levels.

The security provided by a symmetric cryptosystem is not necessarily the same as its key length. The above assumptions hold only if all keys are full-length. Systems of intermediate strength can be obtained by fixing part of the keys. This possibility is not further discussed in this paper (but see Figure 1).

It is assumed that symmetric keys are used for a limited amount of time and a limited encryption volume. Issues related to the limited block length of the DES and its variants are therefore of no concern in this paper.

2.4 Public key security levels. Security levels of public key systems are determined by comparing them to symmetric key security levels. This means that computational and cost equivalence have to be distinguished.

In [7] it is argued that computational and cost equivalence are equivalent measures for the comparison of the security of symmetric systems and ECC. Not explicitly mentioned in [7], and therefore worth mentioning here, is the related fact that the amount of storage needed by the most efficient known attack on ECC (parallelized Pollard rho) does not depend on the subgroup order, but only on the relative cost of processors and storage [21]. In any case, if AES-128 and a certain variant of ECC are computationally equivalent, then they may be considered to be cost equivalent as well.

For the other public key systems, however, there is a gap between computational and cost equivalence. For example, it follows from [7] that AES-128 and about 3200-bit RSA are currently computationally equivalent. With respect to cost equivalence, AES-128 is currently more or less equivalent to 2650-bit RSA. This last estimate depends on an assumption about hardware prices and increases with cheaper hardware. See Section 3 for details. In Sections 3 to 5 both types of equivalence are used to determine public key parameters that provide security equivalent to the symmetric systems. The approach used is based on [7], but entirely geared towards the current application. The results from [7] have been criticized as being conservative [16] – prospective users of AES-192 or AES-256 may be even more conservative as far as security related choices are concerned. The non-ECC entries of most tables consist of two numbers, referring to the cost and computationally equivalent figures, respectively.

3 RSA modulus sizes of matching security

3.1 Current equivalence. Let

$$L[n] = e^{1.923(\log n)^{1/3}(\log\log n)^{2/3}}$$

be the approximate asymptotic growth rate of the expected time required for a factoring attack against an RSA modulus n using the fastest currently known

factoring algorithm, the number field sieve (NFS). This runtime does not depend on the size of the factors of n. It depends only on the size of the number n being factored.

As in [7] actual factoring runtimes are extrapolated to obtain runtime estimates for larger factoring problems. The basis for the extrapolation is the fact that the computational effort required to factor a 512-bit RSA modulus is about 50 times smaller than required to break single DES. With the asymptotic runtime given above it follows that a k-bit RSA modulus currently offers security computationally equivalent to a symmetric cryptosystem of d-bit security and speed comparable to single DES if

$$L[2^k] \approx 50 * 2^{d-56} * L[2^{512}].$$

Furthermore, according to the estimates given in [7], a k'-bit RSA modulus currently offers security cost equivalent to the same symmetric cryptosystem if

$$L[2^{k'}] \approx \frac{50 * 2^{d-56} * L[2^{512}]}{26 * P}.$$

In the latter formula P indicates the (wholesale) price of a stripped down PC of average performance and with reasonable memory. In [7] the default choice $P = 100$ is made. Any other price within a reasonable range of the default choice will have little effect on the sizes of the resulting RSA moduli. See [7, Section 3.2.5] for a more detailed discussion of this issue.

Unlike [7], the relatively speed of the different symmetric cryptosystems under consideration is ignored. The differences observed – comparable implementations of 3DES may be three times slower than single DES, but the AES may be three times faster – are so small that they have hardly any effect on the sizes of the resulting RSA moduli. If desired the right hand sides of the formulas above may be multiplied by v if the symmetric system under consideration is per application v times slower than single DES (using comparable implementations).

3.2 Expected future equivalence. Improved hardware may be expected to have the same effect on the security of symmetric and asymmetric cryptosystems. It may therefore be assumed that over time the relative security of symmetric cryptosystems and RSA is affected only by new cryptanalytic insights that affect one system but not the other.

As far as cryptanalytic progress against symmetric cryptosystems is concerned, it is assumed that they are patched or replaced if a major weakness is found, cf. 2.3.

Progress in factoring, i.e., cryptanalytic progress against RSA, is common. The past effects of improved factoring methods closely follow a Moore-type law [7]. Extrapolation of this observed behavior implies the following. In year $y \geq 2001$ a k-bit RSA modulus may be expected to offer security computationally equivalent to a symmetric cryptosystem of d-bit security if

$$L[2^k] \approx 50 * 2^{d-56+2(y-2001)/3} * L[2^{512}].$$

Cost equivalence is achieved in year y for a k'-bit RSA modulus if

$$L[2^{k'}] \approx \frac{50 * 2^{d-56+2(y-2001)/3} * L[2^{512}]}{26 * P},$$

with P as in 3.1. As in 3.1 effects of the symmetric cryptosystem speed are ignored, and $P = 100$ is a reasonable default choice. For $y = 2001$ the formulas are the same as in 3.1, even though, compared to [7], two years of factoring progress should have been taken into account. Such progress has not been reported in the literature. If progress had been obtained according to Moore's law, its effect on RSA moduli sizes matching the AES would have been between one and two percent, which is negligible.

3.3 Resulting RSA modulus sizes. The formulas from 3.1 and 3.2 with $P = 100$ lead to the RSA modulus sizes in Table 1. The first (lower) number corresponds to the bit-length of a cost equivalent RSA modulus, the second (higher) number is the more conservative bit length of a computationally equivalent RSA modulus. Currently equivalent sizes are given in the row for year 2001, and sizes that can be expected to be equivalent in the years 2010, 2020, and 2030, are given in the rows for those years. It is assumed that factoring progress until 2030 behaves as it behaved since about 1970, i.e., that it follows a Moore-type law. If new factoring progress is found to be unlikely, the numbers given in the row for year 2001 should be used for all other years instead. If factoring progress is expected, but at a slower rate than in the past, one may for instance use the 2010 data for 2020. The data as presented in the table, however, and in particular the computationally equivalent sizes, may be interpreted as 'conservative'. It should be understood that, even for the conservative choices, there is no guarantee that surprises will not occur.

The numbers in Table 1 are not rounded or manipulated in any other way. That is left to the user, cf. [7, Remark 4.1.1]. For the 416-bit RSA modulus cost equivalent in 2001 to single DES, see also Table 2. As an example suppose an

Table 1. Matching RSA modulus sizes.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	416 620	1333 1723	1941 2426	2644 3224	6897 7918	13840 15387
2010	518 747	1532 1955	2189 2709	2942 3560	7426 8493	14645 16246
2020	647 906	1773 2233	2487 3046	3296 3956	8042 9160	15574 17235
2030	793 1084	2035 2534	2807 3408	3675 4379	8689 9860	16538 18260

RSA modulus size has to be determined for an application that uses AES-192 and that is supposed to be in operation until 2020. It follows from Table 1 that RSA moduli should be used of eight to nine thousand bits long. Using RSA moduli of only three to four thousand bits length would undermine the apparently desired security level (namely, higher than AES-128). Five to seven thousand bit RSA moduli would make the public system stronger than AES-128, as desired, but would also make RSA the weakest link if AES-192 lives up to the expectations.

4 RSA factor sizes of matching security

Let

$$E[n,p] = (\log_2 n)^2 e^{\sqrt{2\log p \log\log p}}$$

be the approximate asymptotic growth rate of the expected time required by the elliptic curve method (ECM) to find a factor p of a composite number n (assuming that such a factor exists). This runtime depends mostly on the size of the factor p, and only polynomially on the size of the number n being factored. It follows that smaller factors can be found faster. A regular RSA modulus n has two prime factors of about $(\log_2 n)/2$ bits. In that case the ECM can in general be expected to be slower than the NFS, so the ECM runtime does not have to be taken into account in Section 3. In RSA-MP the RSA modulus has more than two prime factors. This implies that the factors should be chosen in such a way that they cannot be found faster using the ECM than using the NFS. In this section it is analysed how many factors an RSA-MP modulus may have so that the overall security is not affected. It is assumed that the modulus size is chosen according to Table 1, so that the moduli offer security equivalent to the selected symmetric cryptosystem with respect to NFS attacks. It is also assumed that all factors have approximately the same size.

From the definitions of $L[n]$ and $E[n,p]$ it follows that, roughly, the factors p of an RSA-MP modulus n should grow proportionally to

$$n^{(\log n)^{-\frac{1}{3}}}.$$

The size $\log_2 p$ should therefore grow as $(\log_2 n)^{2/3}$, and an RSA-MP modulus n may, asymptotically, have approximately $O((\log_2 n)^{1/3})$ factors. Such asymptotic results are, however, of hardly any interest for this paper.

Instead, given an RSA modulus (chosen according to Table 1) an explicit bound is needed for the number of factors that may be allowed. To derive such a bound the approach from [7] cited in 2.4 is used of extrapolating actual runtimes to derive expected runtimes for larger problem instances. The basis for the extrapolation is the observation that finding a 167-bit factor of a 768-bit number can be expected to require an about 80 times smaller computational effort than breaking single DES ([7, Section 5.9] and [22]). Let n' be an RSA modulus that offers security (computationally or cost) equivalent to a symmetric cryptosystem of d-bit security and speed comparable to single DES (i.e., n' is chosen according to Table 1). An RSA-MP modulus n with smallest prime factor p and with $\log n \approx \log n'$ offers security equivalent to the same symmetric cryptosystem if

$$E[n,p] \geq 80 * 2^{d-56} * E[2^{768}, 2^{167}].$$

Here it is assumed that it is reasonable not to expect substantial improvements of the ECM, and that for application of the ECM itself computational and cost equivalence are the same [16]. Given the least p satisfying the above formula, the recommended number of factors of an RSA-MP modulus n equals $m = [\log n / \log p]$. The resulting numbers of factors are given in Table 2, along with

the bit lengths $\lceil(\log_2 n)/m\rceil$ of the factors, with the computationally equivalent result below the cost equivalent one. Note that $\log_2 p \leq \lceil(\log_2 n)/m\rceil$. For single DES and a cost equivalent RSA modulus in 2001 this approach would lead to a single 416-bit factor, since factoring a composite 416-bit RSA modulus using the ECM can be expected to be easier than breaking single DES. For that reason, that entry is replaced by 'two 217-bit factors'.

Table 2. Number of factors and factor size for matching RSA-MP moduli.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	2 : 217	2 : 667	2 : 971	3 : 882	4 : 1725	4 : 3460
	2 : 310	3 : 575	3 : 809	3 : 1075	4 : 1980	5 : 3078
2010	2 : 259	3 : 511	3 : 730	3 : 981	4 : 1857	5 : 2929
	3 : 249	4 : 489	4 : 678	4 : 890	5 : 1699	5 : 3250
2020	3 : 216	3 : 591	3 : 829	4 : 824	4 : 2011	5 : 3115
	4 : 227	4 : 559	4 : 762	4 : 989	5 : 1832	6 : 2873
2030	3 : 265	4 : 509	4 : 702	4 : 919	5 : 1738	5 : 3308
	5 : 217	5 : 507	5 : 682	5 : 876	5 : 1972	6 : 3044

It can be seen that for a fixed symmetric cryptosystem the number of factors allowed in RSA-MP increases over time. This is mostly due to the fact that the growing moduli sizes 'allow' more primes of the same size, and to a much smaller degree due to the fact that larger moduli make application of the ECM slower.

Almost the same numbers as in Table 2 are obtained if the factor 80 is replaced by any other number in the range $[80/5, 80 * 5]$. Uncertainty about the precise expected behavior of the ECM is therefore not important, as long as the estimate is in an acceptable range.

It may be argued that $E[n, p]$ should include a factor $\log p$. It would make finding larger factors harder compared to the definition used above, and thus would lead to more factors per RSA-MP modulus. For Table 2 it hardly matters. Similarly, the factor $(\log_2 n)^2$ in $E[n, p]$ may be replaced by $(\log_2 n)^{\log_2 3}$ (or something even smaller) if faster multiplication techniques such as Karatsuba (or an even faster method) are used. The effect of these changes on Table 2 is small: for computational equivalence to 2K3DES in 2010 and for cost equivalence to AES-128 in 2020 it would result in three instead of four factors.

4.1 Remark. Although strictly speaking besides the scope of this paper, Table 3 gives the number of factors that may be allowed in RSA-MP moduli of bit lengths 1024, 2048, 4096, and 8192 with the cost equivalent number followed by the computationally equivalent one. It follows, for example, that in the conservative computationally equivalent model one would currently allow three factors in a 1024-bit RSA-MP modulus. But, using less conservative cost equivalence one would, more conservatively, allow only two factors in a 1024-bit modulus (see also Figure 1). This is consistent with the fact that for cost equivalence 1024-bit moduli are considered to be more secure than for computational equivalence: currently just 74 bits for the latter but 85 bits for the former.

Table 3. Number of factors for RSA-MP popular modulus sizes.

Year	1024		2048		4096		8192	
2001	2	3	3	3	3	4	4	4
2010	2	3	3	4	3	4	4	5
2020	3	4	3	4	4	4	4	5
2030	3	5	4	5	4	5	5	5

5 Finite field sizes of matching security

In this section subgroups refer to prime order subgroups of multiplicative groups of finite fields. Public key systems based on the use of subgroups can either be broken by directly attacking the subgroup or by attacking the finite field.

As mentioned in Section 1 the subgroup size will in practice be determined by the hash size. The latter follows immediately from the symmetric cryptosystem choice if the AES is used. Because the subgroup order is prime, the subgroup offers security equivalent to the symmetric cryptosystem as far as direct subgroup attacks are concerned. It remains to select the finite field in such a way that it provides equivalent security as well. That is the subject of this section.

5.1 Fixed degree extension fields. Let p be a prime number and let $k > 0$ be a fixed small integer. The approximate asymptotic growth rate of the expected time to compute discrete logarithms in $\mathbf{F}_{p^k}^*$ is $L[p^k]$, where L is as in 3.1. An RSA modulus n and a finite field $\mathbf{F}_{p^k}$ therefore offer about the same level of security if n and p^k are of the same order of magnitude (disregarding the possibility of subgroup attacks in $\mathbf{F}_{p^k}^*$). It is generally accepted that for such n, p, and k factoring n is somewhat easier than computing discrete logarithms in $\mathbf{F}_{p^k}^*$. For the present purposes the distinction is negligible. Furthermore, it is reasonable to assume the same rate of cryptanalytic progress for factoring and computing discrete logarithms. It follows that Table 1 can be used to obtain matching fixed degree extension field sizes: to find $\log_2 p$ divide the numbers given in Table 1 by the fixed extension degree k.

5.2 Prime fields. It follows from 5.1 that if prime fields are used (i.e., $k = 1$), then conservative field sizes (i.e., $[\log_2 p]$) are given by the numbers in Table 1.

As an example suppose a subgroup and prime field size have to be determined for an application that uses AES-256 and that is supposed to be in operation until 2010. Since SHA-512 will be used in combination with AES-256, the most practical subgroup order is a 512-bit prime. Furthermore, it follows from Table 1 that the prime determining the prime field should be about fifteen thousand bits long. Using eight thousand bits or less would undermine the apparently desired security level (namely, higher than AES-192). A nine to fourteen thousand bit prime would make the public system stronger than AES-192, as desired, but would also make the prime field discrete logarithm the weakest link.

5.3 Extension fields of degrees 2 and 6. LUC and XTR reduce the representation size of subgroup elements by using their trace over a certain subfield

so that the representation belongs to the subfield as well. This does not affect the security and increases the computational efficiency [8, 17].

LUC. LUC uses a subgroup of $\mathbf{F}_{p^2}^*$ of order dividing $p+1$ and traces over $\mathbf{F}_p$. It follows from 5.1 that the size of the prime field $\mathbf{F}_p$ can be found by dividing the numbers from Table 1 by $k = 2$. Table 4 contains the resulting values of $[\log_2 p]$.

XTR. XTR uses a subgroup of $\mathbf{F}_{p^6}^*$ of order dividing $p^2 - p + 1$ and traces over $\mathbf{F}_{p^2}$. The size of the underlying prime field $\mathbf{F}_p$ can be found by dividing the numbers from Table 1 by $k = 6$, resulting in the $[\log_2 p]$-values in Table 5.

Table 4. $[\log_2 p]$ for matching LUC prime fields.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	208 310	667 862	971 1213	1322 1612	3449 3959	6920 7694
2010	259 374	766 978	1095 1355	1471 1780	3713 4247	7323 8123
2020	324 453	887 1117	1244 1523	1648 1978	4021 4580	7787 8618
2030	397 542	1018 1267	1404 1704	1838 2190	4345 4930	8269 9130

Table 5. $[\log_2 p]$ for matching XTR prime fields.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	70 104	223 288	324 405	441 538	1150 1320	2307 2565
2010	87 125	256 326	365 452	491 594	1238 1416	2441 2708
2020	108 151	296 373	415 508	550 660	1341 1527	2596 2873
2030	133 181	340 423	468 568	613 730	1449 1644	2757 3044

Table 6. $[\log_2 p^k]$ for matching small characteristic fields.

Year	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
2001	455 732	1767 2357	2690 3440	3781 4695	10637 12318	22210 24823
2010	592 912	2066 2711	3073 3883	4249 5227	11508 13269	23570 26277
2020	770 1140	2432 3140	3535 4414	4809 5861	12524 14377	25139 27954
2030	977 1398	2835 3608	4037 4986	5412 6539	13594 15539	26771 29694

5.4 Remark. For many of the LUC and XTR key sizes in Tables 4 and 5 there is an integer $e > 1$ such that $(\log_2 p)/e \geq \log_2 q$. This implies that the fields $\mathbf{F}_p$ in LUC and $\mathbf{F}_{p^2}$ in XTR can be replaced by $\mathbf{F}_{\bar{p}^e}$ (LUC) and $\mathbf{F}_{\bar{p}^{2e}}$ (XTR), where $\log_2 \bar{p}^e \approx \log_2 p$ (see [8, Section 6]). Because as a result $\log_2 \bar{p} \geq \log_2 q$, proper $\bar{p}$ and q can still be found efficiently, in ways similar to the ones suggested in [8]. In XTR care must taken that q and $\bar{p}$ are chosen so that q is a prime divisor of $\phi_{6e}(\bar{p})$, the $6e$-th cyclotomic polynomial evaluated at $\bar{p}$, which divides $\bar{p}^{2e} - \bar{p}^e + 1$. In LUC q must divide $\phi_{2e}(\bar{p})$, a divisor of $\bar{p}^e + 1$. With a proper choice of minimal polynomial for the representation of the elements of $\mathbf{F}_{\bar{p}^e}$ (LUC) or $\mathbf{F}_{\bar{p}^{2e}}$ (XTR), this leads to smaller public keys and potentially a substantial speedup (also of the parameter selection). The numbers in Section 6 do not take this possibility into account.

5.5 Small characteristic fields. Let p be a small fixed prime (such as 2), and let $k > 0$ be an extension degree. The approximate asymptotic growth rate of the time to compute discrete logarithms in $\mathbf{F}^*_{p^k}$ for small fixed p is

$$e^{c(\log p^k)^{1/3}(\log\log p^k)^{2/3}}$$

for c oscillating in the interval $[1.526, 1.588]$ (cf. [3]). Since the smallest c leads to the more conservative field sizes, let

$$L'[p^k] = e^{1.526(\log p^k)^{1/3}(\log\log p^k)^{2/3}}.$$

This function is similar to L as defined in 3.1, but has a smaller constant in

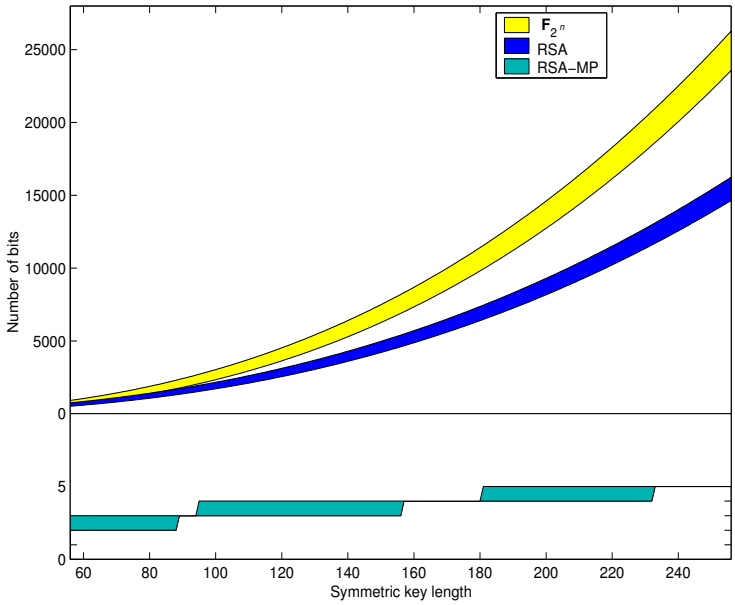

Fig. 1. The sizes from Tables 1 and 6 and the numbers of factors from Table 2 for the year 2010. The shaded areas are bounded from above by the computationally equivalent curves and step function and from below by the cost equivalent ones.

the exponent. This has serious implications for the choice of the field size p^k for small fixed p, compared to the case where k is fixed (as in 5.1). Computing discrete logarithms in $\mathbf{F}_{2^{607}}$ requires an about 25 times smaller computational effort than breaking single DES [19]. It follows that a small fixed characteristic field $\mathbf{F}_{p^k}$ currently offers security computationally equivalent to a symmetric cryptosystem of d-bit security and speed comparable to single DES if

$$L'[p^k] \approx 25 * 2^{d-56} * L'[2^{607}].$$

With respect to cost equivalence and expected future equivalence the same approach as in 3.1 and 3.2 is used: divide the right hand side by $26 * P$ for cost equivalence, and multiply it by $2^{2(y-2001)/3}$ for future equivalence. The resulting values of $[\log_2 p^k]$ for small characteristic fields are given in Table 6 (for $P = 100$); for $p = 2$ the numbers indicate the recommended value for k. Historically, subgroups of multiplicative groups of characteristic two finite fields were mostly of interest because of their computational advantages. Comparing the numbers in Table 1 and Table 6, however, it is questionable if the computational advantages outweigh the disadvantage of the relatively large field size.

6 Performance issues

Assume that public key sizes are chosen according to Tables 1 to 6 to match a symmetric cryptosystem of d-bit security. In this section the impact on public key size overhead and computational requirements is discussed.

6.1 Public key sizes. In Table 7 public key sizes are given for three scenarios. The regular public key refers to all bits contained in the public key. In an ID-based set-up the public key is reconstructed based on the user's identity and an additional number of overhead bits. Refer to [6] for ID-based public key compression for RSA. For subgroup based systems ID-based methods can trivially be designed in almost any number of ways. In a shared public key environment users share a large part of the public key data. In that case only the part that is unique for each user has to be counted.

For subgroup based systems the public key consists of a description of the subgroup, the generator g, its prime order q, and the public point $h = g^s$ (or its trace), where s is the secret key. The generator itself can usually be derived at the cost of an exponentiation of an element with a small representation, and is not counted. In an ID-based system the description of the subgroup and q can be reconstructed from the user's identity and, say, 64 additional bits, which leads to a total public key overhead of 64 bits plus the bits required to describe h. In a shared environment all users use the same g and q, so h is the only part of the public key that is unique for each user.

Fixed degree extension fields are not considered in Table 7, because in that case one may as well use LUC or XTR. The choice of subgroups of multiplicative groups of small characteristic fields is limited. Using such subgroups therefore makes sense only in a context where the public key data, with the exception of the public point h, are shared.

For LUC and XTR the public key sizes follow from [17] and [8]. For ECC the description of the subgroup requires a finite field and an elliptic curve over the field. With d as above, the field and curve take at most $2d$ and $4d$ bits, respectively. About d and $2d + 1$ bits are required for the subgroup order q and the public point h. This leads to $9d + 1$ bits for ordinary ECC, $3d + 65$ bits for ID-based ECC (since the information about q must be present), and $2d + 1$ for shared ECC. For ECC the sizes do not depend on the year. To illustrate the

Table 7. Number of bits required for public key data.

PKC	regular	ID-based	shared
RSA, public exponent e $\log_2 n$ from Table 1 $2\log_2 p = \log_2 n$	$\log_2 e + \log_2 n$	$\log(\frac{1}{2}\log_2 n) + \frac{1}{2}\log_2 n$	n/a
RSA-MP, public exponent e $\log_2 n$ from Table 1 m, $\log_2 p$ from Table 2	$\log_2 e + \log_2 n$	$\log\log_2 p + \frac{m-1}{m}\log_2 n$	n/a
$\mathbf{F}_p$, $\log_2 p$ from Table 1	$2\log_2 p$	$64 + \log_2 p$	$\log_2 p$
$\mathbf{F}_{p^k}$, small p, $\log_2 p^k$ from Table 6	n/a	n/a	$\log_2 p^k$
LUC, $\log_2 p$ from Table 4	$2d + 2\log_2 p$	$64 + \log_2 p$	$\log_2 p$
XTR, $\log_2 p$ from Table 5	$2d + 3\log_2 p$	$64 + 2\log_2 p$	$2\log_2 p$
ECC, $\log_2 p = 2d$	$9d + 1$	$3d + 65$	$2d + 1$

public key size formulas, public key sizes for the year 2010 are given in Table 8, rounded to two significant digits.

6.2 Communication overhead for subgroup based systems. Each message in the Diffie-Hellman key agreement protocol consists of the representation of a subgroup element. The communication overhead per message is given in the last column of Table 7. ElGamal encryption has the same overhead (on top of the length of the message itself). The communication overhead of ElGamal-based message recovery signature schemes is equal to $2d$.

6.3 Computational requirements. In this section the relative theoretical computational requirements are estimated for the most common cryptographic applications of the public key cryptosystems discussed above: encryption, decryption, signature generation, and signature verification. No actual runtimes are given. For software implementations the theoretical estimates should give a reasonable prediction of the actual relative performance. For implementations using dedicated hardware, such as special-purpose exponentiators, all predictions concerning RSA and prime field subgroups are most likely too pessimistic. However, as soon as special-purpose hardware is available for ECC, LUC, or XTR, the relative performance numbers should again be closer to reality.

For subgroup based systems common ElGamal-like schemes are used where decryption and signing each require a single subgroup exponentiation, encryption requires two separate subgroup exponentiations, and signature verification requires the product of two subgroup exponentiations (a 'double exponentiation'). The Diffie-Hellman key agreement protocol has, per party, the same cost as encryption, i.e., two separate subgroup exponentiations.

It is assumed that squaring and multiplication in the finite field $\mathbf{F}_p$ and the ring $\mathbf{Z}/n\mathbf{Z}$ of integers modulo n take the same amount of time if $\log_2 p \approx \log_2 n$. A squaring in $\mathbf{Z}/n\mathbf{Z}$ is assumed to take 80% of the time of a multiplication in $\mathbf{Z}/n\mathbf{Z}$. Basic exponentiation methods are used, i.e., no window tricks. This hardly affects the relative performance. Precomputation of the value g^t with

Table 8. Number of bits of public key data.

PKC	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
regular						
RSA(-MP)	550 780	1600 2000	2200 2700	3000 3600	7500 8500	15000 16000
$\mathbf{F}_p$	1000 1500	3100 3900	4400 5400	5900 7100	15000 17000	29000 32000
LUC	630 860	1700 2100	2400 2900	3200 3800	7800 8900	15000 17000
XTR	370 490	960 1200	1300 1600	1700 2000	4100 4600	7800 8600
ECC	510	860	1000	1200	1700	2300
ID-based						
RSA	270 380	770 990	1100 1400	1500 1800	3700 4300	7300 8100
RSA-MP	270 500	1000 1500	1500 2000	2000 2700	5600 6800	12000 13000
$\mathbf{F}_p$	580 810	1600 2000	2300 2800	3000 3600	7500 8600	15000 16000
LUC	320 440	830 1000	1200 1400	1500 1800	3800 4300	7400 8200
XTR	240 310	580 720	790 970	1000 1300	2500 2900	4900 5500
ECC	230	350	400	450	640	830
shared						
$\mathbf{F}_p$	520 750	1500 2000	2200 2700	2900 3600	7400 8500	15000 16000
$\mathbf{F}_{p^k}$, small p	590 910	2100 2700	3100 3900	4200 5200	12000 13000	24000 26000
LUC	260 370	770 980	1100 1400	1500 1800	3700 4200	7300 8100
XTR	170 250	510 650	730 900	980 1200	2500 2800	4900 5400
ECC	110	190	230	260	390	510

$\log_2 t \approx (\log_2 q)/2$ combined with double exponentiation is used for subgroup based signature generation. For XTR the methods from [18] are used. The LUC and ECC estimates follow from [18, Section 7]. For ECC the time to recover the y-coordinates of subgroup elements is not counted.

The resulting runtime expressions for the four basic cryptographic functions are given in Table 9. Small characteristic fields are not included because the relative speed of $\mathbf{F}_{2^k}$ and $\mathbf{F}_p$ arithmetic is too platform dependent. Despite potential advantages of hardware $\mathbf{F}_{2^k}$-arithmetic, the large value that is required for k may make these fields unattractive for very high security non-ECC cryptographic applications.

As an illustration of the data in Table 9, the relative performance of the cryptographic operations is given in Table 10 for the year 2010, rounded to two significant digits. For Table 10 the time $M(L)$ for modular multiplication of L-bit integers is proportional to L^2. This corresponds to regular hardware implementations. The unit of time is the time required for a single multiplication in $\mathbf{Z}/n\mathbf{Z}$ for a 1024-bit integer n. This arbitrary choice has no influence on the relative performance. For RSA and RSA-MP the sequential ('S') and parallel ('P') performance is given, with the number of parallel processors and the relative parallel runtime separated by a semicolon. RSA encryption and signature verification for $e = 3$ or $e = 2^{17} + 1$ goes about 20 or 3 times faster, respectively, than for a random 32-bit public exponent as in Table 10.

For higher security public key systems other than ECC the finite field and ring sizes get so large that implementation using Karatsuba-like multiplication

Table 9. Number of multiplications in $\mathbf{F}_p$ (unless noted otherwise).

PKC matching symmetric system of d-bit security	encryption	signature verification	decryption	signature generation
RSA, public exponent e $\log_2 n$ from Table 1 $2\log_2 p = \log_2 n$	$1.3\log_2 e$ in $\mathbf{Z}/n\mathbf{Z}$		sequential: $2.6\log_2 p$ 2 in parallel: $1.3\log_2 p$	
RSA-MP, public exponent e $\log_2 n$ from Table 1 m, $\log_2 p$ from Table 2	$1.3\log_2 e$ in $\mathbf{Z}/n\mathbf{Z}$		sequential: $1.3m\log_2 p$ m in parallel: $1.3\log_2 p$	
$\mathbf{F}_p$, $\log_2 p$ from Table 1	$5.2d$	$3.1d$	$2.6d$	$1.6d$
LUC, $\log_2 p$ from Table 4	$6.4d$	$3.5d$	$3.2d$	$1.8d$
XTR, $\log_2 p$ from Table 5	$21d$	$12d$	$10d$	$6d$
ECC, $\log_2 p = 2d$	$36d$	$20d$	$18d$	$10d$

techniques should be worthwhile. In software implementations this can easily be realized. In Table 11 the relative performance for the year 2010 is given using Karatsuba-like modular multiplication. This implies that $M(L)$ is proportional to $L^{\log_2 3}$, as opposed to L^2 as in Table 10. The unit of time in Table 11 is the time required for a single Karatsuba-like multiplication in $\mathbf{Z}/n\mathbf{Z}$ for a 1024-bit integer n. Since this may be different from the time required for a regular 1024-bit modular multiplication (as in Table 10), the numbers in Tables 10 and 11 are not comparable.

As an example of an application of Tables 10 and 11, suppose AES-192 is used in 2010 along with a cost equivalent public key system. With regular (quadratic growth) modular arithmetic, ECC encryption takes time equivalent to about 970 regular multiplications modulo a 1024-bit modulus. This can be expected to be about twice faster than RSA encryption (with a 32-bit public exponent), and about six times faster than XTR encryption. But with Karatsuba-like arithmetic, RSA encryption takes time equivalent to about 960 Karatsuba multiplications modulo a 1024-bit modulus (but using a 7400-bit modulus). This can be expected to be about 1.5 times faster than ECC encryption, and about six times faster than XTR. For decryption, however, RSA is substantially slower than both ECC and XTR for either type of arithmetic, even if RSA-MP is used on four parallel processors.

6.4 Parameter selection. For all public key systems except ECC, parameter selection is dominated by the generation of the primes defining the moduli, finite fields, and subgroup orders. For each L-bit prime to be generated, the generation time is proportional to $M(L)L^2$. A more precise runtime function depends on a wide variety of implementation choices that are not discussed here. Obviously, parameter selection for high security RSA, prime field, or LUC based systems will be slow compared to RSA-MP and, in particular, XTR.

For systems based on a subgroup of $\mathbf{F}_{p^k}$ for fixed small p public key data are usually shared (except for the public point h). For such systems the speed of parameter selection is therefore not an important issue.

Table 10. Relative performance using regular arithmetic for the year 2010.

PKC	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
			$\log_2 n$			
RSA(-MP)	520 750	1500 2000	2200 2700	2900 3600	7400 8500	15000 16000
			$\log_2 p$			
$\mathbf{F}_p$	520 750	1500 2000	2200 2700	2900 3600	7400 8500	15000 16000
LUC	260 370	770 980	1100 1400	1500 1800	3700 4200	7300 8100
XTR	90 130	260 330	370 450	490 590	1200 1400	2400 2700
ECC	112	190	224	256	384	512
encryption (with $\log_2 e = 32$ for RSA and RSA-MP)						
RSA(-MP)	11 22	93 150	190 290	340 500	2200 2900	8500 10000
$\mathbf{F}_p$	75 160	1100 1800	2700 4100	5500 8000	53000 69000	270000 340000
LUC	23 48	340 550	820 1300	1700 2500	16000 21000	84000 100000
XTR	8 17	120 200	290 450	610 890	5800 7600	30000 37000
ECC	24	120	190	290	970	2300
decryption						
RSA (S)	43 130	1100 2300	3300 6200	7900 14000	130000 190000	970000 1300000
RSA (P)	2 : 22 2 : 65	2 : 560 2 : 1200	2 : 1600 2 : 3100	2 : 3900 2 : 7000	2 : 63000 2 : 95000	2 : 490000 2 : 660000
RSA-MP (S)	43 57	500 580	1400 1500	3500 3500	32000 30000	160000 210000
RSA-MP (P)	2 : 22 3 : 19	3 : 170 4 : 150	3 : 480 4 : 390	3 : 1200 4 : 870	4 : 7900 5 : 6100	5 : 31000 5 : 43000
$\mathbf{F}_p$	37 77	550 900	1300 2000	2700 4000	26000 34000	140000 170000
LUC	11 24	170 280	410 630	840 1200	8100 11000	42000 51000
XTR	4 9	61 99	150 230	300 450	2900 3800	15000 18000
ECC	12	59	96	140	490	1100
signature generation						
RSA (S)	43 130	1100 2300	3300 6200	7900 14000	130000 190000	970000 1300000
RSA (P)	2 : 22 2 : 65	2 : 560 2 : 1200	2 : 1600 2 : 3100	2 : 3900 2 : 7000	2 : 63000 2 : 95000	2 : 490000 2 : 660000
RSA-MP (S)	43 57	500 580	1400 1500	3500 3500	32000 30000	160000 210000
RSA-MP (P)	2 : 22 3 : 19	3 : 170 4 : 150	3 : 480 4 : 390	3 : 1200 4 : 870	4 : 7900 5 : 6100	5 : 31000 5 : 43000
$\mathbf{F}_p$	23 48	340 550	820 1300	1700 2500	16000 21000	84000 100000
LUC	6 13	93 150	230 340	460 680	4400 5800	23000 28000
XTR	2 5	36 58	85 130	180 260	1700 2200	8700 11000
ECC	7	32	53	79	270	630
signature verification (with $\log_2 e = 32$ for RSA and RSA-MP)						
RSA(-MP)	11 22	93 150	190 290	340 500	2200 2900	8500 10000
$\mathbf{F}_p$	44 92	660 1100	1600 2400	3300 4800	31000 41000	160000 200000
LUC	13 26	190 300	450 690	930 1400	8900 12000	46000 57000
XTR	5 10	71 120	170 260	350 520	3400 4400	17000 21000
ECC	13	65	110	160	530	1300

Table 11. Relative performance using Karatsuba arithmetic for the year 2010.

PKC	DES	2K3DES	3K3DES	AES-128	AES-192	AES-256
			$\log_2 n$			
RSA(-MP)	520 750	1500 2000	2200 2700	2900 3600	7400 8500	15000 16000
			$\log_2 p$			
$\mathbf{F}_p$	520 750	1500 2000	2200 2700	2900 3600	7400 8500	15000 16000
LUC	260 370	770 980	1100 1400	1500 1800	3700 4200	7300 8100
XTR	90 130	260 330	370 450	490 590	1200 1400	2400 2700
ECC	112	190	224	256	384	512
encryption (with $\log_2 e = 32$ for RSA and RSA-MP)						
RSA(-MP)	14 25	79 120	140 190	220 300	960 1200	2800 3300
$\mathbf{F}_p$	99 180	940 1400	1900 2700	3500 4800	23000 29000	90000 110000
LUC	40 72	380 560	800 1100	1500 2000	9400 12000	37000 44000
XTR	23 41	220 320	450 630	820 1100	5400 6600	21000 25000
ECC	60	240	360	510	1500	3100
decryption						
RSA (S)	76 200	1300 2400	3200 5500	6800 11000	74000 110000	430000 560000
RSA (P)	2 : 38 2 : 99	2 : 630 2 : 1200	2 : 1600 2 : 2700	2 : 3400 2 : 5600	2 : 37000 2 : 53000	2 : 220000 2 : 280000
RSA-MP (S)	76 100	660 790	1700 1800	3600 3700	25000 25000	100000 130000
RSA-MP (P)	2 : 38 3 : 34	3 : 220 4 : 200	3 : 560 4 : 460	3 : 1200 4 : 930	4 : 6200 5 : 4900	5 : 20000 5 : 26000
$\mathbf{F}_p$	49 88	470 690	970 1400	1800 2400	12000 14000	45000 53000
LUC	20 36	190 280	400 560	730 980	4700 5800	18000 22000
XTR	12 21	110 160	230 320	410 560	2700 3300	10000 12000
ECC	30	120	180	260	730	1500
signature generation						
RSA (S)	76 200	1300 2400	3200 5500	6800 11000	74000 110000	430000 560000
RSA (P)	2 : 38 2 : 99	2 : 630 2 : 1200	2 : 1600 2 : 2700	2 : 3400 2 : 5600	2 : 37000 2 : 53000	2 : 220000 2 : 280000
RSA-MP (S)	76 100	660 790	1700 1800	3600 3700	25000 25000	100000 130000
RSA-MP (P)	2 : 38 3 : 34	3 : 220 4 : 200	3 : 560 4 : 460	3 : 1200 4 : 930	4 : 6200 5 : 4900	5 : 20000 5 : 26000
$\mathbf{F}_p$	30 54	290 420	600 840	1100 1500	7100 8800	28000 33000
LUC	11 20	110 160	220 310	400 540	2600 3200	10000 12000
XTR	7 12	63 93	130 180	240 320	1600 1900	6100 7200
ECC	17	65	100	140	400	840
signature verification (with $\log_2 e = 32$ for RSA and RSA-MP)						
RSA(-MP)	14 25	79 120	140 190	220 300	960 1200	2800 3300
$\mathbf{F}_p$	59 110	560 820	1200 1600	2100 2900	14000 17000	54000 63000
LUC	22 40	210 310	440 610	800 1100	5200 6400	20000 24000
XTR	13 24	130 190	260 370	480 650	3100 3900	12000 14000
ECC	33	130	200	280	800	1700

ECC parameters can be found in expected polynomial time. Nevertheless, even for security equivalent to 2K3DES the solution is not yet considered to be sufficiently practical for systems with non-shared keys. The slow growth of the parameter sizes implies, however, that if a satisfactory solution is found for current (relatively low) security levels, then the solution will most likely also work fast enough for very high security levels. For ECC over fields of characteristic two this goal is close to being achieved [4].

7 Summary of findings

Matching AES-192 or AES-256 security levels with public key systems requires public key sizes far beyond anything in regular use today. For instance, to match the security of AES-192 with RSA, it would be prudent to use moduli of about 7000 bits. But given current resources, the overall practicality of RSA with such moduli is questionable. Encryption and signature verification are faster than for any other system if the public exponent is small, but the modulus itself may be prohibitively large. RSA-MP fares a little better. But even if fully parallelized it is still relatively unattractive. An interesting observation is that computationally equivalent RSA-MP moduli often allow more factors than the (smaller) cost equivalent ones, and may thus attain greater decryption and signature generation speed (at the cost of a higher level of parallelism).

The unattractive sizes of RSA moduli of high security levels is entirely due to the number field sieve. If it had not been invented, and the asymptotically slower quadratic sieve factoring algorithm would still be the fastest factoring algorithm, then at least until 2030 RSA moduli of 2048, 4096, and 8192 bits would be good matches for AES-128, AES-192, and AES-256, respectively. But, it could have been worse too: if the special number field sieve would apply to RSA moduli, then RSA moduli would have to be chosen according to Table 6 instead of Table 1, i.e., considerably larger.

Compared to RSA and RSA-MP, subgroups of prime fields have the same size problem. They are much slower for encryption and signature verification. Decryption and signature generation is competitive only in environments where RSA and RSA-MP cannot be parallelized. Furthermore, subgroups of prime fields are consistently outperformed by LUC and XTR. So, unless second and sixth degree extension fields turn out to be less secure than currently believed, subgroups of prime fields are not competitive.

Similarly, LUC is consistently outperformed by XTR^2. Unless a dramatic breakthrough occurs in the fixed degree extension field discrete logarithm problem, XTR is a good choice if one insists on using a non-ECC subgroup public key system. It has the additional advantages that parameter selection is easy and that current special purpose RSA modular multipliers (that can handle public moduli up to, say, 1024 bits) may be used even for very high security applications (possibly using Remark 5.4). The latter is also possible for LUC (if Remark 5.4

[2] However, for LUC it is in general faster to test if a value is correctly formatted, i.e., if it is the trace of a proper subgroup element. Refer to [9] for details.

is used), may be possible for RSA-MP, but is out of the question for RSA or prime field subgroups.

Overall, ECC suffers the smallest performance degradation when moving to very high security levels. Generation of ECC public keys in a non-shared set-up remains problematic, for all security levels. If that is not a concern, and barring cryptanalytic progress affecting the elliptic curve discrete logarithm problem, the choice is obvious.

For current security levels, i.e., comparable to 1024-bit RSA, the choice is between RSA, RSA-MP, XTR, and ECC and will mostly depend on the application. For current higher security levels, comparable to 2048-bit RSA, the theoretical performance gap between ECC and the other public key systems already becomes noticeable, with only XTR still within range of ECC. However, hardware accelerators are currently available for 2048-bit RSA and RSA-MP, but not for other security equivalent public key systems. So, for the next few years RSA and RSA-MP will still be the methods of choice in many practical circumstances where security equivalent to 2048-bit RSA is required. This may change radically if new types of hardware accelerators are developed. And even if that does not happen, it will change eventually, i.e., for higher security levels, because special purpose hardware cannot beat the asymptotics.

Disclaimer. The contents of this paper are the sole responsibility of the author and not of his employer. The author does not accept any responsibility for the use of the material presented in this paper. Despite his academic involvement with XTR, the author does not have any financial or other material interests in any of the cryptosystems discussed in this paper.

Acknowledgments. The author thanks Eric Verheul and Mike Wiener for their many insightful comments on earlier versions of this paper and Martijn Stam for his assistance with Figure 1.

References

1. I. Blake, G. Seroussi, N. Smart, *Elliptic curves in cryptography*, Cambridge University Press, 1999.
2. H. Cohen, A. Miyaji, T. Ono, *Efficient elliptic curve exponentiation using mixed coordinates*, Proceedings Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 51-65.
3. D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. Inform. Theory 30 (1984) 587-594.
4. R. Harley, Rump session presentations at Eurocrypt 2001 and Crypto 2001; data available from argote.ch/Research.html.
5. P.C. Kocher, *Breaking DES*, RSA Laboratories' Cryptobytes, v. 4, no 2 (1999), 1-5; also at www.rsasecurity.com/rsalabs/pubs/cryptobytes.
6. A.K. Lenstra, *Generating RSA moduli with a predetermined portion*, Proceedings Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 1-10.
7. A.K. Lenstra, E.R. Verheul, *Selecting cryptographic key sizes*, to appear in the Journal of Cryptology; available from www.cryptosavvy.com.
8. A.K. Lenstra, E.R. Verheul, *The XTR public key system*, Proceedings of Crypto 2000, LNCS 1880, Springer-Verlag 2000, 1-19; available from www.ecstr.com.

9. A.K. Lenstra, E.R. Verheul, *Fast irreducibility and subgroup membership testing in XTR*, Proceedings PKC 2001, LNCS 1992, Springer-Verlag 2001, 73-86; available from www.ecstr.com.
10. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
11. National institute of standards and technology, *Digital signature standard*, FIPS Publication 186-2, February 2000.
12. National institute of standards and technology, //csrc.nist.gov/encryption/aes/.
13. National institute of standards and technology, //csrc.nist.gov/cryptval/shs.html.
14. R.L. Rivest, A. Shamir, L.M. Adleman, *Cryptographic communications system and method*, U.S. Patent 4,405,829, 1983.
15. B. Schneier, *Applied cryptography*, second edition, Wiley, New York, 1996.
16. R.D. Silverman, *A cost-based security analysis of symmetric and asymmetric key lengths*, RSA Laboratories Bulletin 13, April 2000.
17. P. Smith, C. Skinner, *A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms*, Proceedings of Asiacrypt '94, LNCS 917, Springer-Verlag 1995, 357-364.
18. M. Stam, A.K. Lenstra, Speeding up XTR, Proceedings Asiacrypt 2001, Springer-Verlag 2001, this volume; available from www.ecstr.com.
19. E. Thome, *Computation of discrete logarithms in* $\mathbf{F}_{2^{607}}$, Proceedings Asiacrypt 2001, Springer-Verlag 2001, this volume.
20. P.C. van Oorschot, M.J. Wiener, *A known-plaintext attack on two-key triple encryption*, Proceedings Eurocrypt'90, LNCS 473, Springer-Verlag 1991, 318-325.
21. M.J. Wiener, personal communication, August 2001.
22. P. Zimmermann, personal communication, 1999.