Recent developments in cryptography

Arjen K. Lenstra

Citibank, N.A., Technische Universiteit Eindhoven, 1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.
arjen.lenstra@citicorp.com
May 2001

Abstract. Modern information protection methods can only be effective if an almost impossibly wide range of conditions is met. Proper cryptography is one of these conditions. This article reviews some recent cryptographic development, but focuses on understanding the cryptographic decision making process.

1 Introduction

This article reviews some recent trends and developments in cryptography. Understanding the impact of the latest advances in cryptography is necessary to achieve and maintain adequate information security, but it is by no means sufficient: cryptography is not the solution to anybody's security problems, it is just one of its many ingredients.

Cryptography is a mathematical discipline and as such a subject where non-specialists can easily feel overwhelmed. All too often cryptographic tools are described in terms that are beyond the grasp of even well-informed security managers. Also if plain language is used, it is not always obvious how it should be interpreted. Examples abound. Application of a cryptographic method that is heralded as 'provably secure' does in general not lead to a system that cannot be broken into. Neither does the latest announcement of a 'broken cryptographic standard' imply immediate practical vulnerabilities.

The target audience of this paper consists of higher level security managers and officers who have a broad range of decision making responsibilities. The purpose is to help them get a proper understanding of the practical implications and relevance of cryptographic developments that have recently caught wider-than-usual attention. Thus, for instance, an attempt is made to explain what 'provably secure' may stand for, and why usage of 'broken' systems does not necessarily lead to immediate disasters. Also, attention is paid to the reasons behind the upcoming abolishment of the Data Encryption Standard (DES) and the subsequent introduction of the Advanced Encryption Standard (AES).

The list of non-cryptographic security related issues not treated in this paper is virtually endless. At the highest level, and often overlooked, is the definition of a suitable trust model. Without it the entire concept of 'information security' does not make sense, and auditing becomes impossible. At the opposite end of the spectrum one finds the users and the password selection problem. A satisfactory

solution to the latter requires cooperation of the entire user community – not the least reason why security is an elusive objective. Enforcing the multitude of security related policies is probably the least enviable task of the security manager. This article offers no solution to this problem – or to any problem for that matter – only the observation that users tend to follow the rules better if they understand what the rules are supposed to be good for – just as the sequel is meant to make some of those arcane cryptographic schemes more palatable.

A conservative security design should avoid any method that has not yet been subjected to many years of public scrutiny. Newfangled or proprietary schemes are therefore not discussed; only middle-of-the-road and well-publicized cryptographic issues are discussed in this article. Many of these methods have been standardized, usually after a lengthy period of public review. This is just one of the reasons why compliance with the 'standards' can in general be recommended. Other reasons include compatibility, off-the-shelf availability – the average company cannot be expected and should not attempt to write its own cryptography or security related software and system integration – and rapid dissemination of and recovery from security problems. Unfortunately, also the choice between different standards-compliant vendors can be bewildering. This article should provide some help to properly evaluate vendor proposals and to pose the right questions while deciding what cryptography to use.

This article is organized as follows. Section 2 explains that modern cryptography combines two rather different techniques, namely symmetric and asymmetric cryptographic methods, and lists the currently most popular methods for both. Some of the recent developments regarding symmetric and asymmetric cryptography are further discussed in Sections 3 and 4, respectively. Background for most material presented in this article and references can be found in [12, 15].

2 Modern Cryptography

Classical cryptography is mostly concerned with confidentiality of communications, achieved by so-called *symmetric cryptosystems*. In a symmetric cryptosystem each pair of communicating parties agrees in advance upon a particular method of encryption and decryption. Often all participants use the same fast textual transformation method. But each pair of participants shares a certain unique and hard to guess 'key' that, when used in combination with the transformation method, results in encryptions that are inaccessible to anyone who does not have the proper key, even if the transformation method itself is known. The latter assumption is known as *Kerkhoffs' assumption*: the strength of a system should never be based on secrecy of the transformation method.

The requirement that each pair of communicating parties shares the same symmetric secret key makes it impossible to directly apply classical cryptography to internet based communications: how is someone supposed to agree on a shared secret key with any other party on the internet that one happens to be communicating with? This problem is solved using *public key cryptography* by means of so-called *asymmetric cryptosystems*, as briefly described below. Al-

though much slower than symmetric cryptosystems, they can be used, among others, to establish a shared symmetric secret key between any two parties. Next, once the shared key is established, the parties may use a symmetric cryptosystem to exchange their messages.

Kerkhoffs' assumption is still fundamental in modern cryptography, despite the fact that it aims for more than just confidentiality. The main goals of modern cryptography are usually summarized as follows:

Confidentiality Making it impossible for unauthorized parties to extract anything useful from information, either stored or in transit.

Authentication Providing proper identification of the source of information.

Integrity Providing assurance that no unauthorized changes have been made to the information.

Non-repudiation Making it impossible for the source of the information to later deny its role.

These goals are achieved by a wide variety of cryptographic protocols that are all based on some combination of the following three basic primitives:

Symmetric Cryptosystems where two parties share a secret key and use it in conjunction with (assumed to be publicly known) encryption and decryption methods. These primitives are closest in flavor to the methods used in classical cryptography.

Examples

- Block ciphers such as the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES).
- Stream ciphers.
- Message authentication codes (MACs).
- Keyed (pseudo)random number generators.

Asymmetric Cryptosystems where each party A is identified by a particular *public key* P_A that may be known to everyone and that was computed as a non-invertible function of the party's *private* or *secret key* S_A. Anyone with access to P_A can use a publicly known transformation method along with P_A to encrypt information, but the corresponding decryption can be computed only by the party that knows S_A (i.e., supposedly, A). Alternatively, A may use S_A to sign a digital document, resulting in a digital signature that can be verified by anyone with access to P_A. These at first sight rather counter-intuitive primitives were first formulated in the 1970s – they distinguish modern cryptography from classical cryptography. It should be noted that the non-invertibility of the function from S_A to P_A, i.e., the fact that it is impossible to derive S_A from P_A, refers to a practical computational impossibility, not to mathematical impossibility: it should be practically infeasible or intractable to compute S_A given P_A.

Examples

- Factorization based systems, in particular RSA (named after its inventors Rivest, Shamir, and Adleman).

– Discrete Logarithm systems such as the Diffie-Hellman key agreement protocol, the ElGamal encryption and signature protocols, the U.S. Digital Signature Standard (DSS), all of them either using traditional multiplicative groups of finite fields (or subgroups thereof à la Schnorr) or using elliptic curves (ECC, for 'Elliptic Curve Cryptography').

Keyless Functions such as hash functions, hardware random number generators, and other auxiliary functions.

Examples of current hash functions are MD5, RIPEMD, and the U.S. Secure Hash Algorithm (SHA-1) and its forthcoming extensions SHA-256, SHA-384, and SHA-512.

Cryptographic protocols generally consist of a sequence of steps involving any number of any of the primitives mentioned above. If one of the steps uses the RSA asymmetric cryptosystem, the protocol is said to be based on RSA; protocols may be based on more than one asymmetric cryptosystem, though this is rather exceptional. How the primitives work together to realize cryptographic protocols that achieve the above mentioned goals is not described in this article. Instead it concentrates on some of the recent developments related to some of the primitives mentioned above (DES, AES, RSA, and ECC) and protocols based thereon.

Obviously, there are many more recent developments in this area than can be covered in this article. An example of a subject not touched upon here is given by the so-called environmental or side-channel attacks: how well is a system protected against attacks that do not try to take advantage of possible weaknesses in the regular communications to and from a processor or device carrying out cryptographic functions, but that try to discover secret information from changes – or by causing changes – in their environment: changes in power consumption or data files, variations in radiation, effects on the computation (and the outcome) by subjecting the device to extreme temperatures or radiation, etc. The effectiveness of such attacks varies from system to system (either in hardware or in software). Before making a decision on what system to employ one needs to be well aware of these vulnerabilities.

3 Symmetric Cryptosystems

The most important recent development concerning symmetric cryptosystems is without question the demise of the DES and the introduction of the AES. Now that the U.S. National Institute for Standards and Technology (NIST) has announced that Rijndael, the Belgian submission, is its final choice, the new standard will soon be finalized. Most likely good hardware implementations will be available within the next couple of years, thereby enabling incorporation of the AES in any newly built system requiring symmetric encryption capabilities. It is less obvious to what extent usage of the DES in existing systems should be replaced by the AES. In particular it is not immediately clear that this should be done in existing stand-alone systems without compatibility requirements with any other existing or future systems. This issue is discussed in this section.

The DES uses a 56-bit key to transform a 64-bit block of plaintext to a 64-bit block of ciphertext. The AES uses 128-bit plaintext and ciphertext blocks and a key size of either 128, 192, or 256 bits. Because an additional key bit is supposed to make a symmetric cryptosystem twice stronger, the least secure version of the AES is designed to be

$$\underbrace{2 \times 2 \times \ldots \times 2}_{128-56} = 2^{72} > 10^{21}$$

times more secure than the DES. At this point it remains to be seen if that is indeed the case.

Despite more than two decades of cryptanalysis, the most efficient attack to retrieve a single message that is encrypted using the DES is exhaustive key search, i.e., trying all 56-bit keys until the right decryption is found. Thus, it can be stated without exaggeration that the DES was extremely well designed. Nevertheless, these days it is no longer considered to be sufficiently secure, even for applications that require only a moderate level of security. This does not mean to imply that anyone can instantaneously break any DES-encrypted message: in 1997 a software based DES-key search took 4 months on the internet [13], and in 1998 a special purpose hardware device was built, at the cost a quarter million dollars, that retrieves a DES key in, on average, less than 5 days [10]. These figures provide upper bounds for the security of the DES – there is little doubt that well-funded agencies can find a DES key much quicker and that they were able to do so long before 1997.

The fact that it has been 'broken', whatever that exactly means, does not imply that the DES is immediately totally worthless. It is still quite suitable, if nothing better is easily available, for the protection of non-critical applications with a very short life span. More interestingly, the security can be improved substantially by applying the DES several times, with several different keys. Double application of the DES does not add any security: due to a so-called 'meet-in-the-middle' attack two-key double-DES can be broken in the same time as one-key single-DES (i.e., DES), although much more memory is required for the attack. But if the DES is used three times, with at least two different keys, then it is generally believed that additional security is obtained compared to one-key single-DES.

The most popular security enhanced version of the DES is two-key triple-DES: three rounds of DES, where the same key is used in the first and last round, but with a different key for the second round. Also, for compatibility with one-key single-DES, the second round uses the inverse of the DES, which is as hard to break as the DES itself. Two-key triple-DES supposedly attains strength equivalent to that of an ideal 80-bit key symmetric cryptosystem, at the actual cost of two 56-bit keys and three DES applications. Two-key triple-DES is easily available in hardware, despite the fact that one of the DES designers referred to it as an 'historical error' [5]. Three-key triple-DES is much better, relatively speaking: it is believed to attain 112-bit key strength, at the cost of three 56-bit keys and three DES applications. Efficient hardware implementations of it are beginning to become available.

According to [3, 11] symmetric cryptosystems of 90 bits strength provide sufficient security for the foreseeable future. And 112-bit security is effectively available from three-key triple-DES, albeit somewhat inefficiently at the cost of three 56-bit keys and three applications of the DES. So why does anyone need the AES? That depends. In the first place, three consecutive applications of the DES was felt to be a too inefficient way to achieve 112 bits of security, let alone 128 bits, and certainly too inefficient to allow real time software decryption of encrypted video streams. A more serious obstacle to general acceptance and continued future use of repeated DES modes is, however, that the block length remains 64 bits. That implies that after encryption with the same key of about $\sqrt{2^{64}} = 2^{32}$ blocks, i.e., 32 Gigabytes, the same encryption blocks will begin to show up, leading to undesirable vulnerabilities. With current data rates it is hard to imagine that 32 Gigabytes will be sent during a single transmission (i.e., without key refreshment). For future data rates this may not be so far-fetched.

If these reasons do not apply, and all that is needed is a security upgrade of some stand-alone legacy system that already contains a satisfactory one-key single-DES implementation, then migration to three-key triple-DES is probably much cheaper than, and essentially as effective as, replacing the DES by the AES. It is conceivable that this applies to, for instance, customer banking transactions, since most of the financial transactions involved are low volume and not time-critical. If, however, speed is or will be an issue, or if large transmissions are anticipated, or if compatibility may become important, then it is probably best to upgrade to the AES as soon as adequate software and hardware implementations become available. In any case, continued use of one-key single-DES for anything of even the slightest importance is irresponsible. It can no longer be recommended.

A final note in this section on hash functions. As a rule of thumb, for general applications the length of the hash function should be at least twice the length of the symmetric cryptosystem key. Thus, the DES with 56-bit keys can be used alongside any hash function of length $2 \times 56 = 112$ bits or more, such as MD5 (of length 128) or SHA-1 (of length 160). But if three-key triple-DES (with keys of effectively 112 bits) or the 128-bit version of the AES are going to be used, then MD5 or SHA-1 no longer provide adequate (i.e., matching) security for all applications, and the 256-bit hash function SHA-256 will have to be used. Similarly, the 192-bit or 256-bit variants of the AES require usage of SHA-384 and SHA-512, respectively. Usage of MD5 should be discouraged anyhow, irrespective of its length, as it is generally believed to be insufficiently secure [7].

4 Asymmetric Cryptosystems

4.1 Breaking 512-bit RSA

In the RSA cryptosystem the public key of each participant contains a so-called RSA modulus, a large integer that consists of the product of two prime factors.

If any of the two prime factors of a participant's public RSA-modulus can be found, then the private key of that participant can be found, and the system is considered to be broken. If the primes are properly chosen (i.e., large enough), then finding them given only their product (the RSA-modulus) is believed to be a computationally infeasible task. To make the system secure the primes must therefore be chosen sufficiently large. On the other hand, large primes imply a large RSA-modulus, which leads to substantial computational overhead when using the RSA system.

Thus, in RSA there is a trade-off between security and efficiency: on the one hand moduli must be large for security, on the other hand small moduli are preferred for efficiency. How large they have to be depends on the speed of so-called factorization algorithms. In the late 1970s factorization algorithms were barely able to factor integers consisting of 160 bits (about 50 decimal digits), and 320-bit RSA moduli were considered to be secure. Faster computers and better factorization algorithms, each to the same degree, made it possible to factor 320-bit RSA moduli in the late 1980s, and to break the famous 426-bit (129-digit) 'RSA challenge' in 1994. Although at their time these factorizations counted as very substantial achievements, right now factoring 320-bit numbers is a triviality, and for 426-bit numbers it is a rather common computation. It follows that, in order to get any degree of security whatsoever, RSA-moduli must be much larger than 426 bits. Note that RSA moduli are much larger than typical symmetric cryptosystem keys, such as 56-bit DES keys. This is because the degree of security obtained depends in an entirely different way on the number of bits. See [11] for an extensive discussion of this often confusing issue.

In the fall of 1999 it was announced that a 512-bit RSA modulus was broken [4]. This was done by a large team of experts on several hundreds of computers scattered over almost all continents and required more than 4 months. No surprise therefore that the resulting announcement that '512-bit RSA is broken' was met with considerable skepticism. If breaking a single 512-bit RSA modulus requires such an enormous effort, surely the average 512-bit RSA modulus must be reasonably secure? It better be, since according to unsubstantiated sources, the majority of the 'secure' internet traffic was protected by 512-bit RSA moduli.

Does one have to be concerned about announcements of this sort and in the present case upgrade RSA system implementations to a higher level of security? Or can they be ignored as sensationalism of a bunch of publicity-hungry scientists? It is certainly true that the average 512-bit RSA key runs no immediate risk to be broken. But announcements of this sort serve as an early warning that the security is no longer what it was, and that conservative designs should be upgraded as soon as feasible. In the case of factorization efforts (breaking RSA moduli) this is particularly true. Once a certain bit-size has been successfully attacked because all related problems (usually quite many) have been overcome by the leading experts in the field, the cryptanalytic technology is available, and can be used by anyone else. Generally speaking, it takes only a few years before the general public can achieve factorizations equivalent to record-breaking

factoring work, at the cost of only a moderate computational effort, relatively speaking (cf. Remark 4.1.2). For 512-bit RSA moduli this point was proved convincingly when, within a year of the first 512-bit RSA factorization, a team of students that were not at all specialized in factoring, managed to break another 512-bit RSA modulus, mostly using the same software that was used for the earlier 512-bit effort, in a fraction of the wall-clock time (but using more computing time) [17].

Thus, in case of 512-bit RSA the message is clear, despite the initial skepticism: 512-bit RSA keys should no longer be used for anything of even the slightest importance. And actually, they should not have been used since the early 1990s. Right now one should begin looking beyond 1024-bit RSA moduli.

Remark 4.1.1 Factoring an RSA-modulus is a generic way to attack any cryptographic protocol whose security relies on the RSA cryptosystem. If successful it breaks the protocol, and does not use or need any protocol specific properties: knowledge of the (public) RSA-modulus suffices for a factoring attempt. Generic attacks are, so the speak, the heaviest type of artillery one can use against a protocol, because they directly attack the underlying cryptosystem. Properly designed protocols are not susceptible to any attacks other than generic ones, i.e., it should be impossible to exploit specific protocol properties that are not related to the underlying cryptosystem. See also Subsection 4.2.

Remark 4.1.2 Comparison of the computational effort of factoring a 512-bit RSA modulus and retrieving a 56-bit DES key shows that the latter is about 50 times harder! It is just because factorization algorithms are so much more complicated to understand, to implement, and to run compared to key-retrieval methods, that it took longer for 512-bit RSA to be broken than 56-bit DES. It should be understood, however, that the successful attacks referred to in this paper are the published attacks in the open literature – it is conceivable, and indeed most likely, that similar feats were achieved much earlier in the closed community.

4.2 Provable Security

The design of cryptographic protocols used to be a happy-go-lucky activity: just design a protocol, publish it, and wait for it to be broken. If it does not get broken, it is apparently secure. These days lack of successful cryptanalysis is no longer counted as a strong argument in favor of a protocol. New protocols are generally no longer accepted for publication without some type of assurance of their strength. This is of course a most welcome change. But it is also a risky one, because it may be misleading to unsuspecting outsiders such as the average user: protocols that are alleged to be 'provably secure', as advertised for many newly designed protocols, may be better than the 'old' ones about which nothing could be proved, but provably secure they are certainly not. In this subsection this issue is discussed.

In 1993 the first version of the popular RSA encryption standard PKCS #1 was published [14]. PKCS #1 is a cryptographic protocol that is based on the

RSA cryptosystem. Among others, it describes the precise format of the data to be encrypted, at first sight consisting of rather innocuous-looking restrictions on size and format of the padding bits. Nothing really could be proved about the security of this standard, except of course that it could be broken by factoring the RSA modulus (cf. Remark 4.1.1), but neither did it look suspicious. Until, in 1998, Daniel Bleichenbacher announced that the encryption protocol could be broken by means of a so-called *adaptive chosen ciphertext attack*, where an attacker may submit its own carefully designed ciphertexts and have them decrypted by the owner of the secret key [2]. It was shown that, in case of 1024-bit RSA, about 1 million chosen ciphertexts suffice to let the attacker decrypt a message of its choice.

Evidently, this is not good. But, users of the first version of PKCS #1, when evaluating the impact of this attack on their set-up, should first analyze if chosen ciphertexts attacks can be mounted against their system to begin with, and in the second place if a huge number of messages can realistically be submitted at all. In many cases there will be no reason for panic. Nevertheless, the message is clear: version 1 of PKCS #1 turned out to be substantially less secure than anticipated, and protocols must be designed much more carefully.

What this 'much more carefully' means is already explained, to a certain extent, in Remark 4.1.1: a protocol based on cryptosystem X is properly designed only if it can be proved that attacking the protocol requires breaking X. Thus, a protocol is called 'provably secure' if the only successful attack against it is a generic attack on the underlying cryptosystem. This still leaves open what the attack model is. Some protocols may be provably secure under the relatively weak notion of 'chosen plaintext attacks', others may be provably secure under 'chosen ciphertext attacks', which is a stronger notion.

Thus, 'provable security' actually means something that is quite different from provable security. It means that the security can provably be *reduced* to the security of the underlying cryptosystem (i.e., the protocol can be broken only if the underlying cryptosystem can be broken), and that the reduction holds under a certain attack model – fine points that are often conveniently overlooked by the marketing department. Users of a provably secure cryptosystem still have to evaluate how realistic an attack model is, and if in their set-up stronger attacks may be feasible. And, more importantly, they have to hope that the underlying asymmetric cryptosystem is indeed as secure as it is believed to be. Unfortunately, provably secure asymmetric cryptosystems have not been found yet: the common belief in the security of RSA, Discrete Logarithms in finite fields, ECC, etc., is almost exclusively based on the observation that after, allegedly, years of research no fast and realistic cryptanalytic tools have been published, not on proofs substantiating the difficulty of real-life problems. The restriction to 'realistic tools' is important, because if so-called *quantum computers* can be built then most currently popular asymmetric cryptosystems are easily breakable [16]. According to [8]: 'Building a quantum computer is one of physics' greatest challenges'. At this point it is unclear if it will ever be possible to build one that is large enough to pose any danger to current asymmetric cryptosystems.

What is the implication for users of version 1 of PKCS #1 who feel vulnerable because of Bleichenbacher's attack? These days several 'provably secure' alternatives exist. For instance, version 2 of PKCS #1 is not susceptible to the attack and is provably secure, if one is willing to believe:

- the so-called *random oracle model*, something on which the jury is still out but which 'seems to provide security in practice' [1]. This is yet another vague statement, but at least a vague statement that this protocol has in common, provably, with many other protocols;
- the assumption that RSA is non-invertible, an assumption that may be stronger than the assumption that factorization is difficult.

This holds in the strongest attack model, i.e., chosen ciphertext attacks, thus giving the strongest possible security assurance. It may be interesting to note that version 2 of PKCS #1 dates from 1994, but that its 'proof of security' is very recent and will appear only this year [1].

Another cryptosystem that is provably secure against chosen ciphertext attacks is the Cramer-Shoup cryptosystem [6]. It is reasonably efficient (though less so than version 2 of PKCS #1) and it does not rely on the random oracle model. Instead it only needs a good hash function, an assumption that is generally found to be fairly mild and quite acceptable. Unlike PKCS #1 the underlying cryptosystem is not RSA, but something that is known as the *Decision Diffie-Hellman* problem. This means that, in principle, the Cramer-Shoup cryptosystem can be implemented using any group where the Decision Diffie-Hellman is hard, thus providing a wide variety of possible implementations.

There is more to this story. With the rising popularity of Elliptic Curve Cryptography someone may have decided back in 1998, after the publication of Bleichenbacher's attack, to use the group of an elliptic curve over some finite field in an implementation of the Cramer-Shoup cryptosystem. Generating random elliptic curves with the right properties is, however, not an easy task. Thus, the decision may have been made to use a particular type of elliptic curves for which it can easily be ascertained that all relevant properties are satisfied. The most obvious example of such curves is given by so-called supersingular elliptic curves. The resulting implementation of the Cramer-Shoup cryptosystem is provably secure against chosen ciphertext attacks, and relies for its security on a good hash function and the difficulty of the Decision Diffie-Hellman problem in the group of a supersingular curve over a finite field. Unfortunately, the latter does not amount to much: it was found only very recently that the Decision Diffie-Hellman problem in the group of a supersingular curve over a finite field can be solved efficiently [9, 18]. Thus, the new 'provably secure' system turns out to be much less secure (i.e., trivially breakable) than the 'broken' first version of PKCS #1 that it meant to replace.

This does not at all imply that 'provably secure' systems are bad. Neither does it imply that Elliptic Curve Cryptography is bad, though the above example shows that there are sometimes unpleasant surprises. The above illustrates that a proper interpretation of even such simple terms as 'provably secure' is far beyond, in level of detail and in variety of implications, of what can be expected

of someone who is supposed to have a broad overview of security policies and decisions – there is simply too much one should be aware of. It helps to understand that security is, and always will be, a moving target. Unbiased expert opinions can be very helpful to recognize the constantly changing problems and to suggest appropriate solutions. The problem is not so much obtaining these opinions. The problem that turns out be the hardest to overcome is admitting that something may be wrong. If that fear cannot be conquered, security will remain elusive.

References

1. M. Bellare, *Provably-secure public-key cryptosystems*, invited lecture at PKC 2001, Cheju Island, South Korea, February 2001.
2. D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1*, Proceedings Crypto'98, LNCS 1462, Springer 1998, 1-12.
3. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, *Minimal key lengths for symmetric ciphers to provide adequate commercial security*, www.bsa.org/policy/encryption/cryptographers_c.html, January 1996.
4. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al., *Factorization of a 512-bit RSA modulus*, Proceedings Eurocrypt 2000, LNCS 1807, 1-17, Springer 2000.
5. D. Coppersmith, invited lecture at Crypto 2000, Santa Barbara, August 2000.
6. R. Cramer, V. Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Proceedings Crypto'98, LNCS 1462, Springer 1998, 13-25.
7. H. Dobbertin, lecture at 'Public key cryptography and computational number theory', Warsaw, Poland, September 2000.
8. *Quantum dreams*, The Economist, March 10, 2001, 81-82.
9. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, Proceedings ANTS IV, LNCS 1838, 358-394, Springer 2000.
10. P.C. Kocher, *Breaking DES*, RSA Laboratories' Cryptobytes, v. 4, no 2 (1999), 1-5; also at www.rsasecurity.com/rsalabs/pubs/cryptobytes.
11. A.K. Lenstra, E.R. Verheul, *Selecting cryptographic key sizes*, to appear in the Journal of Cryptology; available from www.cryptosavvy.com.
12. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
13. www.rsa.com and www.rsasecurity.com.
14. RSA Data Security, Inc., *PKCS #1: RSA encryption standard*, Redwood City, 1993, Version 1.5.
15. B. Schneier, *Applied cryptography*, second edition, Wiley, 1996.
16. 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.
17. Simon Singh's cipher challenge, www.simonsingh.com/cipher.htm.
18. E.R. Verheul, *Evidence that XTR is more secure than supersingular elliptic curves*, Proceedings Eurocrypt 2001, to appear, Springer 2001.

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