A Survey of Available Cryptosystems

Clifford Bergman*
Department of Mathematics, Iowa State University, Ames, Iowa 50011

March 27, 2002

1 Introduction

Cryptographic techniques have become a standard component in many communications settings. There is now a wide variety of tools, both in hardware and software, providing encryption for data security and digital signatures for authentication. Almost everyone involved in data transfer encounters these tools in the course of their work. End users may wish to send electronic mail or store data locally without fear that this information can be accessed by unauthorized persons. Hardware and software developers may similarly wish to incorporate encryption or authentication schemes in their products.

Because of the dizzying array of options, many users are confronted with questions they may not be prepared to answer: which tools, and more specifically, which algorithms, are most appropriate for a given task?

In this article, we survey some commonly available cryptographic algorithms, discuss their security and make recommendations as to which algorithms are best suited to which applications.

1.1 Some Key Notions

In any comparison of cryptosystems, it is important to bear in mind several central ideas. Most important is the distinction between *symmetric-key* cryptography and *public-key* cryptography. Very briefly, in a symmetric system, the encryption and decryption keys are the same. (Or if not the same, each is easily derivable from the other.) By contrast, the defining characteristic of a public-key system is that the *de*cryption key can not be easily computed from knowledge of the *en*cryption key.

*this whitepaper originally written under contract to ICS Edge in Des Moines, Iowa

The remarkable properties of public-key systems come at a price: they are slower and more expensive than are most symmetric-key systems. According to a 1989 study [5] the fastest hardware implementation of RSA encrypts at a rate of about 30 kilobits/second (with a 512 bit modulus). At that time, there were dedicated DES chips on the market with a throughput of almost 1 gigabit/second [16, page 279]. When implemented in software, the differences between the two systems are not as great, but still significant. Software implementations of DES operate about 100 times faster than comparable implementations of RSA.

In practice, most packages that support public-key operations actually operate in a hybrid mode that utilizes both a public-key and a symmetric cipher. Let us suppose that Alice wishes to initiate a secure communication with Bob. Alice first selects a random key to the symmetric cipher, called a *session key.* She looks up Bob's public key in a directory and encrypts the session key using Bob's public key. This encrypted key is then sent to Bob, who can decrypt it using his private key. All subsequent communication between Alice and Bob is encrypted via the symmetric cipher using the session key.

1.2 Block vs. Stream Ciphers

A second distinction we wish to make is between a block cipher and a stream cipher. In some sense the distinction is a bit artificial since a block cipher can be used as a stream cipher and visa-versa. But most ciphers are designed with the expectation that they will be used one way or the other.

A block cipher encrypts data in blocks, typically of either 64 or 128 bits. If the same block occurs several times in the source, the encryption of the block will be the same each time. The encryption algorithm involved in a block cipher is usually quite complicated. Block ciphers are used when high and long-lasting security is required.

By contrast, a stream cipher processes data in very small units, often one bit at a time. The encryption of a bit will depend 50% on its (unencrypted) value and 50% on its location in the input stream. Generally speaking, a stream cipher offers a lesser degree of security but much higher speed than even the fastest symmetric block ciphers. Stream ciphers are most appropriate for real-time encryption, such as for cell phone transmissions or pay-for-view movies.

1.3 Types of Attacks

Thirdly, we discuss the types of cryptanalytic attacks we wish to resist. At the low end are *ciphertext only* attacks. That is, the opponent obtains some ciphertext, but knows little or nothing about the corresponding plaintext. Obviously, we expect

every cryptosystem to be strongly resistant to such an attack since we are typically transmitting encrypted material over insecure channels.

The next more dangerous attack is a *known-plaintext* attack. In this scenario, the opponent is in possession of several blocks of plaintext together with the corresponding ciphertext. We assume that the opponent has no control over the nature of the plaintext blocks. The known-plaintext attack may seem far-fetched, but in fact, experience has proved that a determined opponent can almost always mount such an attack. For example, most file formats have a standard header, or standard keywords at predictable points in a file. If the opponent intercepts a file of a known file type, he can frequently deduce the header and obtain pairs of plaintext/ciphertext blocks. There is a general consensus in the cryptographic community that every modern cipher should be highly resistant to a known-plaintext attack.

Note that a "brute force" attack (see below) on a cipher is generally assumed to be of the known-plaintext variety. In this scenario we know several pairs (p_1, c_1), (p_2, c_2),..., (p_n, c_n) of blocks, with c_i the encryption of p_i using the (unknown) key k. In an attempt to determine k, we encrypt p_1 using every possible key until we obtain c_1. The keys that work are then checked against the remaining pairs $(p_2, c_2), \ldots, (p_n, c_n)$, until we are left with only one candidate, which must necessarily be k. It is generally sufficient to take $n = 2$ for this attack.

The third category of attack with which we will be concerned is a *chosen-plaintext* attack. Here, not only does the opponent have access to several pairs of the form (p, c) with c the encryption of p under an unknown key, but he gets to choose the values of the p blocks. Such a situation may occur if the cyptanalyst has temporary access to the encryption device. For example, a lost or stolen smart card might be tamper-proof (so that the built-in key can not be accessed directly), but could still be attacked in this way.

Note that in particular, every public-key cryptosystem must be resistant to a chosen-plaintext attack. After all, the encryption key is assumed to be known to all, so our enemy can encrypt as much plaintext as he or she wants in an effort to determine our private key.

1.4 Brute-force attacks

Perhaps a discussion of the notion of a "brute-force attack" is in order. A better term might be "black-box attack". In these attacks, the cipher is considered to be a black-box. That is, we are able to monitor what goes in and what comes out of the cipher, but we do not attempt to use any knowledge of the algorithm employed by the cipher. The attack boils down to simply trying every possible key until the correct one is found.

There is actually a spectrum of black-box attacks. At one end are attacks that

require a great deal of time, but little space. At the other end of the spectrum are attacks that are fast but highly space-intensive. Let us assume for a moment we are attacking a cipher with an n-bit key. Given a pair (b, c) with c the encryption of b under an unknown key, we can compute $E_k(b)$ for every possible key k until we obtain c. The k that achieves this is probably the unknown key. This known-plaintext attack uses very little space but requires 2^n encryptions in the worst case. Let us also remark that this mode of attack parallelizes very well. One need only partition the key space and set a different processor loose on each member of the partition. No inter-process communication is required.

At the opposite extreme, we could fix a block b and precompute $E_k(b)$ for every key k, storing the results in a table. Note that the table only has to be computed once. Now when we obtain the encryption of b under some unknown key, we can quickly find the ciphertext block in our table and read off the key. The attack is very fast, but requires enough space for the table. This approach should be considered a chosen-plaintext attack since it requires the encryption of a specific block b.

It is also possible to design black-box attacks that reside somewhere in the middle of the spectrum. For example in [7], Hellman presents an attack that requires $2n \cdot 2^{2n/3}$ bits of storage and approximately $2^{2n/3}$ encryptions for an n-bit key. At CRYPTO 2001, J.-J. Quisquater demonstrated Hellman's attack on 40-bit DES. Using nothing but a laptop computer (with the tables precomputed), Quisquater found six separate keys in under 10 minutes. [Note that 40-bit DES is not a contrived cipher. Until U.S. regulations were revised, 40-bit keys were the maximum permitted for export. To this day there are undoubtedly millions of copies of Netscape and Windows 2000 in use that rely on 40-bit encryption.]

A related idea is the *meet-in-the-middle* attack of Merkle and Hellman [13]. Let us suppose that E and F are two not necessarily related ciphers with key sizes of n and m bits, respectively. We shall assume that the block sizes of E and F are the same. Given keys k and l for these two ciphers, we obtain the *cascade cipher* $F_l(E_k(b))$. In other words, we first encrypt the plaintext block b with the cipher E and then encrypt the result with F. The key for this new cipher consists of the pair (k, l), of size $n + m$ bits. Thus, the black-box attacks discussed above require either time or space on the order of $2^{n+m} = 2^n \cdot 2^m$. However, the meet-in-the-middle attack uses approximately $2^n + 2^m$ encryptions and a table big enough to hold 2^n blocks of text.

Finally, let us mention a technique called *whitening.* Suppose that E is a cipher with a block size of n bits and a key size of m bits. We can define a new cipher EX by the rule $EX(b) = k_2 \oplus E_k(b \oplus k_1)$. In this definition, k is a key for the cipher E (m bits) while k_1 and k_2 are bit strings of length n. The triple (k_1, k, k_2) constitutes the key for EX. Note that EX can be seen as a cipher with a key size of $2n + m$ bits. It is proved in [8], that if the most efficient attack on E is a black-box

attack as described above, then the same is true for EX. Thus whitening seems to be a very effective way of lengthening a key without significantly impacting the performance or security of the underlying algorithm.

2 Symmetric Block Ciphers

In this section we shall survey several publicly available symmetric block ciphers from the standpoint of security. Our conclusions can be stated quite simply. None of the ciphers we consider: variants of DES and the five AES finalists, exhibit any intrinsic weakness. The only known or anticipated attack on any of these ciphers is some version of a brute-force attack described above.

2.1 DES in single and multiple modes

The venerable digital encryption standard (DES) uses a 56-bit key and a 64-bit block. DES is in widespread use throughout the world. Despite a tidal wave of suspicion that the algorithm was designed with a "back door", even the most successful attacks on DES are only slightly superior to brute-force. Differential cryptanalysis (Biham and Shamir, [2]) is a chosen-plaintext attack that can find a DES key using 2^{47} encryptions, given 2^{47} chosen plaintext-ciphertext pairs. Linear cryptanalysis (Matsui, [11]) cracks DES after 2^{43} encryptions with 2^{43} known plaintext-ciphertext pairs. The enormous amount of data required by these attacks renders them useless in practice.

Ultimately, it was the ever-increasing processing speed of modern computers that proved to be the downfall of DES. In 1999 a special-purpose DES-cracking machine was built that finds a key, using brute-force, in approximately 24 hours, [6]. It is now generally accepted that because of its small key, DES is no longer secure. However the DES *algorithm* is still believed to be a good one. At this time, no other cipher has a track record that inspires the kind of confidence granted DES.

Unfortunately, it does not seem to be possible to scale the DES algorithm to use a longer key. On the contrary, there is evidence that the design of DES is quite fragile [2, 9, 12]. The only way to increase its lifespan is by cascading multiple copies of the cipher. These composite ciphers are subject to the meet-in-the-middle attack discussed above.

Double DES is a cascade of two copies of DES using two independent keys. While the key size is ostensibly 112 bits, the meet-in-the-middle attack requires only 2^{57} encryptions of single DES and a table of 2^{60} bytes. The number of encryptions is certainly within the range of current technology. The size of the table might put the attack out of reach for the moment, but not for long. Surely double

DES will be vulnerable within the next decade.

Triple DES comes in two varieties. EDE encryption uses two keys, k_1 and k_2. A 64-bit block b is encrypted as $E_{k_1}(D_{k_2}(E_{k_1}(b)))$, that is, the block is first encrypted using the key k_1, then decrypted using k_2 and finally encrypted again using k_1. On this arrangement, the meet-in-the-middle attack turns out to be slightly worse than brute-force. However, there is a different attack [19] that exploits the EDE design. Given a set of t known plaintext-ciphertext pairs, this attack finds the keys (k_1, k_2) after computing $2^{120}/t$ (on average) encryptions.

True three-key DES (known as 3DES in the literature) is a straightforward cascade using three encryptions and three independent keys. (Usually it is implemented with a decryption as the second stage of the cipher.) The meet-in-the-middle attack can be mounted on 3DES. It requires 2^{113} encryptions and a table containing 2^{60} bytes. No better attack is known. The large number of encryptions puts the attack out of reach for the foreseeable future. 3DES is a Federal Information Processing Standards Algorithm, and, according to [14], it "will remain a FIPS-approved algorithm for the foreseeable future."

One caution on the implementation of 3DES (or any cascade cipher) is in order. (Single) DES is often implemented using one of a number of feedback functions, such as cipher block chaining mode. A discussion of feedback modes is out of the scope of this document. Let us just say that feedback throws some additional obstacles in the path of the cryptanalyst, and also allows for a certain amount of parallelism in the design. However, in [1], Biham makes an exhaustive analysis of common feedback modes and demonstrates that every one of them introduces insecurities into a cascade cipher and renders it only slightly stronger than a single-stage cipher. In other words, all stages of a cascade cipher should be implemented in "electronic code book mode", that is, without feedback.

There are several supplementary points to make about triple DES. First, it uses a very long key: 168 bits, but affords only 112 bits of protection. Second, with a block size of 64 bits, it is conceivable that one could mount a "dictionary attack" against the cipher. That is, one could attempt to accumulate the encryption of each of the 2^{64} possible plaintext blocks under some unknown key. Although this would not allow us to determine the key, we could still read every message encrypted with this key. Furthermore, the short block size renders 3DES useless as a hash function, a topic we do not consider further in this paper. Thirdly, the cascade construction requires the computation of three encryptions, in series. Thus, 3DES runs relatively slowly in practice. For all of these reasons, we are inclined to recommend that new applications employ a more modern algorithm with a larger key and block size, rather than 3DES. However, there is no reason to believe that existing 3DES-applications will become insecure any time soon.

2.2 The Advanced Encryption Standard

In 1997 the National Institute of Standards and Technology (NIST) began the process of selecting a cryptosystem to replace DES as a Federal standard for the encryption of unclassified data. This culminated in October 2000 with the designation of a cipher called Rijndael as the new Advanced Encryption Standard (AES). Rijndael was one of five "finalists" in the selection process, the others being MARS, RC6^{TM}, Serpent and Twofish. Each of these ciphers has a block size of 128 bits and each can be designed to use a key of either 128, 192 or 256 bits. All five algorithms have a fine pedigree and are available on a royalty-free basis.

The selection process was highly public and feedback was solicited from cryptographers world-wide. There seems to be almost unanimous satisfaction with NIST's conduct of the search and the eventual decision. A detailed report on the entire process including the criteria for evaluation is available in [14].

Based on three years of analysis, NIST concluded that all five cryptosystems seem to be secure. The report asserts "there are no known security attacks on any of the five finalists, and **all five algorithms appear to have adequate security for the AES**" [14, page 88]. In fact, the selection of Rijndael seems to have been based primarily on its speed and versatility, rather than on any security superiority to the other four finalists.

One can reasonably conclude that any of these five algorithms can be utilized in a security system with great confidence. Nevertheless, I recommend that Rijndael be used for several reasons. First, it will offer a high degree of interoperability since virtually all new cryptographic products will support Rijndael. Second, cryptanalysis of Rijndael will only increase now that it has been designated as the AES. Any successful attacks, or even potential attacks, will be widely publicized, and users will react accordingly.

It may be desirable to maintain a second algorithm as a backup in the unlikely event that Rijndael is compromised. For extremely sensitive data, one may wish to double-encrypt with Rijndael as the first stage and some other cipher as the second. Any of the other four finalists would make a good choice here. So would triple-DES. However, for the overwhelming majority of applications, Rijndael will be more than adequate and should provide many years of security.

2.3 Other Algorithms

There are several encryption products available, both commercial and otherwise, that offer a bewildering choice of ciphers. The atmosphere in cryptographic circles is such that if any well-known cipher is compromised, that information is rapidly disseminated world-wide. Weak algorithms can be quickly removed from

distribution. None of the ciphers in common use have any known vulnerabilities. However, many of them have not received enough attention from the cryptographic community to be considered especially secure. We briefly survey a few of the more common ciphers here. By a n/k-bit cipher we shall mean a cipher with a block size of n bits and a key of k bits.

IDEA is a $64/128$-bit cipher designed in 1992. It has received quite a bit of attention because of its selection as the symmetric cipher in the first widely distributed version of PGP (pretty good privacy [15]). Although no attacks on IDEA have ever been discovered, there are patent issues that resulted in the cipher being dropped from most open-source encryption products.

CAST refers to a series of ciphers developed since 1993. Over the years, the various incarnations of CAST have been cryptanalyzed. Vulnerabilities have been found in some of the early versions, but the most recent ones seem to be secure. A 128/128-bit version was submitted as an AES candidate, but was not one of the final selections. Although no significant vulnerabilities have been found, the NIST team judged the algorithm to be slower than many of the other algorithms and to be unsuitable for smartcard implementations.

GOST is a $64/256$-bit cipher created in the former Soviet Union. The design is somewhat unorthodox by western standards. In addition to the 256 bits of key information, GOST requires an additional 512 bits of "secondary key" data. As far as I can tell, the secondary key can not be chosen at random and, in fact, the security of the resulting cipher may depend on the specific choice of secondary key. There is little publicly available cryptanalysis of GOST at this time.

SAFER SK-128 is a $64/128$-bit cipher created by J. Massey for the Cylink Corp. It has been adapted for use by the government of Singapore. This version of SAFER seems to be quite secure. However, the user must be careful. Two earlier versions, SAFER S-64 and SAFER S-128 are not considered secure.

Blowfish is a 64-bit block cipher with a variable-length key, designed by Bruce Schneier of Counterpane Systems. Surprisingly, despite the rather high profile of its inventor, Blowfish has received very little in the way of serious cryptanalysis. Twofish, a descendant of Blowfish, was one of the AES finalists.

Perhaps we should also mention Skipjack. This algorithm was designed by the National Security Agency for use in the Clipper and Capstone encryption chips and is now used in the "Fortezza" PC-cards. Skipjack is a 64/80-bit cipher that has withstood quite a bit of analysis both in the public domain and presumably, within the NSA. Most uses of Fortezza require that the key be escrowed with the federal government.

	DES	3DES	Rijndael	Other AES	IDEA	SAFER SK-128
Email		★	★	☆	☆	☆
Local file encryption			★	★	☆	☆
Streaming video			★	★		
Smart cards	☆		★	Serpent		

★ Best choice ☆ Acceptable choice

Table 1: Recommended symmetric encryption algorithms

2.4 Recommendations

There is no single algorithm that is best for all applications. NIST, faced with the requirement of picking a single best "all-around cipher" chose Rijndael. However the informed user is not constrained by this requirement, and can instead tailor his or her choice to the application at hand.

Electronic mail. For secure email, neither encryption speed nor block size are major concerns. In this scenario one should choose an algorithm with a large key and a proven track record. Triple DES has both. Rijndael is another good choice, and the truly paranoid could double-encrypt with both. For those with a "legacy" system, IDEA should provide adequate security.

Hard drive encryption and streaming video. For these applications, a 64-bit block size is a liability, and speed is at a premium. Triple DES is inadequate here. Rijndael, or any of the other AES finalists would be appropriate. For ultra-high speed, one may turn to a stream cipher, albeit with a possible degradation of security.

Smart cards. The particular characteristics of a smart card impose yet a different set of requirements, including processor power, memory use and vulnerability to power analysis. Of the AES candidates, Rijndael and Serpent were judged particularly suitable for this application.

3 Public-key Ciphers

Public-key cryptography is a general term that covers both encryption techniques and digital signatures. As a rule, any public-key encryption system can be used, with minor modifications, as a signature scheme, and visa-versa. For our purposes, it is convenient to lump the two notions together. One point on which these two techniques differ is lifetime. Most often, it is only necessary to keep a data file

encrypted for a short time: a few months or a year at most. However, a signature may need to be secure for a very long time. Home mortgages, for example, can have a thirty year lifetime. When evaluating a public-key cryptosystem, this time frame should be taken into account.

Unlike symmetric ciphers, public-key ciphers are generally based on established mathematical principles. Consequently, they tend to have a great deal of structure which can be exploited both by the cryptographer (attempting to establish a lower bound on security) and the cryptanalyst (attempting to compromise the cipher). There has been slow but steady progress on the mathematical problems that underlie most public-key cryptosystems. For this reason, evaluating the security of such a cipher involves predicting the future advances in certain areas of mathematics.

One interesting wrinkle, is that, unlike most symmetric systems, the common public-key cryptosystems such as RSA and El Gamal can be used with keys of arbitrary length. Thus as raw computing power increases, one need not discard the cipher entirely, but merely move on to a longer key.

There have been several attempts to project past advances into the future. Two recent discussions are those of Lenstra and Verheul [10] and Silverman [17]. The Lenstra-Verheul article is particularly interesting because of its detailed analysis of the many different assumptions inherent in any quantification of security.

The focus in the Lenstra-Verheul article is on the two fundamental problems around which most public-key systems are designed: integer factorization and computation of discrete logarithms. By making explicit assumptions such as the rate of increase in computational power and combining this with established asymptotic analysis of existing attacks, the authors compute time frames in which various cryptosystems can be considered secure. They go to great length to explicate their specific assumptions (many of which are open to debate), and even explain how to recompute the final results after modifying those assumptions.

Lenstra-Verheul observe that there has been steady improvement in the *techniques* used for factoring (as opposed to simply building faster computers that implement existing techniques more quickly). For this reason, they project that the minimum necessary key-size for RSA will increase at a faster rate than would be dictated by Moore's law. They apply similar standards to traditional discrete-log systems such as El Gamal. However, it should be noted that the algorithms are less effective on discrete-log problems, and consequently, the Lenstra-Verheul model may be understating the security of El Gamal vis-a-vis RSA.

Furthermore, Lenstra-Verheul assert that there has been no progress in algorithms to attack either subgroup discrete-log systems (such as the digital signature algorithm) or elliptic curve systems since 1978, with the exception of a technique to parallelize Pollard's rho method (the rho method is the 1978 algorithm). Based

on this, the model predicts that key sizes for these cryptosystems can grow very slowly for the foreseeable future.

One criticism of the Lenstra-Verheul model is its reliance on asymptotic upper bounds for the *time* required to solve a problem. This focus is customary in theoretical computer science, and the tools are fairly well developed. However some have argued that rather than time, the *cost* of a computation is the critical measurement in cryptanalysis. The cost of a computation is generally considered to be proportional to the product of the time and memory required to perform that computation. (Even the definition of cost is open to argument. For example, a network of off-the-shelf PC's might be far cheaper than a single special-purpose piece of hardware, even if the 'time $\times$ memory' requirement of the latter is smaller.)

This is the point of view taken by Silverman. By modeling the cost of the most powerful factoring methods, Silverman predicts that the key-requirements of RSA will grow much more slowly than do Lenstra and Verheul. On the other hand, Silverman does not seem to allow for the possibility that any genuinely new progress in factoring (or discrete log) technology will appear any time soon. One interesting point to note is that both models arrive at similar conclusions for the security of elliptic curve systems.

3.1 Systems based on integer factorization

By far, the most common public-key cryptosystem based on factorization is RSA, named for its inventors, R. Rivest, A. Shamir and L. Adelman. RSA was the first published example of a public-key cipher, and has received more study over the years than any other system. Although there is no proof of the fact, there is overwhelming evidence that compromising a general RSA key is no easier than is factoring the public modulus.

Although factoring algorithms continue to improve, overall the improvement has been gradual and largely linear. One possible exception is a recent announcement of D. Bernstein which sketches a special-purpose hardware design that has the potential to triple the length of an integer that can be factored for a given cost. Whether such a computer is genuinely practical remains to be seen.

In addition to those that involve the underlying factoring problem, several other attacks on RSA have been discovered. For many years it was assumed that, because it was public, the encryption exponent could be chosen to facilitate encryption without any compromise in security. For example, the choice of both 3 and 17 allow for very efficient encryption. However, a series of discoveries, culminating in [4], demonstrate that all or most of the private key can be easily computed from such small public encryption exponents. For a very readable discussion of this and similar attacks, see D. Boneh's article in [3]. There are numerous exploits against

improper implementations of RSA. See [16, Sec. 19.3] for a summary.

There are several other public-key cryptosystems that are based on factorization, such as the Rabin system and the Blum-Goldwasser system. To our knowledge, none of these systems have been implemented in a commercial product.

3.2 Systems based on discrete logarithms

There are numerous public-key schemes whose security is based on the difficulty of computing discrete logarithms. In its simplest form, this problem asks, given integers g, b and p, for a solution to the equation $g^x \equiv b \pmod{p}$. Such an x is called a *discrete log of b to the base g modulo p.* This problem seems to be of comparable difficulty to the integer factorization problem when p is a large prime, g a primitive element modulo p and b is arbitrary.

The El Gamal cryptosystem is based on discrete logs in the form described above. El Gamal can be used for both encryption and digital signatures. Progress in the discrete log problem roughly parallels that of factoring. In fact, there is a surprising similarity between the methods used for the two problems. Like RSA, there are also some attacks on the implementation of El Gamal. In particular, El Gamal requires a source of random numbers. A poor random number generator could conceivably compromise the system.

Although it is not technically a cryptosystem, the Diffie-Hellman key exchange is based on the discrete log problem. (Unlike El Gamal, it does not require random numbers.) For many purposes, Diffie-Hellman can be used in place of a public-key algorithm when negotiating a session key for a symmetric cipher.

There are other ciphers that are based on variants of the discrete log problem. (To be precise, they are based on discrete logs in groups other than the group of integers modulo p.) The Digital Signature Algorithm (DSA) is one example. Although it is officially used only for signatures, it can also be used for encryption, see [16, pg. 490]. As we discussed earlier, the most powerful method for solving the discrete log problem, called index-calculus, seems to be less effective when applied to the variant used in DSA. This suggests that it is possible to get a comparable level of security with a shorter key than would be used in El Gamal or RSA. On the other hand, the current Digital Signature Standard put forth by NIST restricts the field size of DSA to 1024 bits, which many consider to be too short for long-term use.

Elliptic curve cryptosystems (ECC) are based on discrete logarithms in still other groups. Although elliptic curves have been studied for more than a century, our understanding of these objects is still much less developed than is our knowledge of the integers. So far, no one has been able to use index-calculus to attack elliptic curve ciphers. Whether this is due to an intrinsic property of elliptic curves or merely to our ignorance is an open question. The best we can say is that *at this*

	RSA	El Gamal DSS	ECC
Email	1024	1024	
Long-life signatures	3000	3000	225
Smart Cards			128

Table 2: Recommended modulus length (in bits) for public key ciphers

time ECCs offer the same level of security as RSA but with a considerably shorter key and correspondingly faster performance. However, there does seem to be a greater potential that a significant mathematical breakthrough will render existing ECCs insecure than would be true for RSA. Elliptic curve systems are available from several vendors.

3.3 Other public-key systems

There are several other cryptosystems in the literature. The McEliece system is based on a difficult problem in the field of error-correcting codes. There are no commercial implementations of the cipher, probably because it requires a very long public key. The NTRU system is based on a combination of a factoring problem and a lattice-reduction problem. This system is quite new and there have been a series of advances in the analysis of the system, some leading to compromises. NTRU is available commercially from the company of the same name.

3.4 Recommendations

Most applications of public-key cryptography involve powerful computers (such as a modern PC) on both ends of the transaction. These include electronic mail, secure-shell logins and digital signatures. In this context, there is no reason to be stingy with the key-size. I recommend taking a conservative course and sticking with either RSA or a conventional discrete-log algorithm such as El Gamal or DSA. Use a modulus of at least 1024 bits. 1500 bits would be even better.

For very long-term signature integrity, the Lenstra-Verheul model suggests that the modulus should be on the order of 3000 bits. (However, those authors warn against extending their model too far into the future.) Unfortunately, the Digital Signature Standard does not allow for such a long key using conventional discrete logs. One alternative is to use RSA signatures instead, or simply ignore the Federal standards and use the DSA anyway. Users who wish to (or need to) be in compliance with the DSS will have to use the new elliptic curve DSA, see [18]. A field

size of at least 225 bits is recommend. For example, Curves P-256, P-384, K-283 or B-283 from FIPS 186-2 would be adequate. In my opinion, the reliance on elliptic curves does entail some additional risk in that the potential for a mathematical breakthrough concerning elliptic curves is higher than it is for discrete logs over the integers.

There are applications in which we can not afford to be so extravagant with our key sizes. These primarily involve smart cards, which are limited in both storage and processing power. Under these circumstances elliptic curve cryptosystems offer the best security potential. The reservations I expressed above are not applicable here, since the lifetime of a smart card is relatively short—typically only a couple of years. A field size of 128 bits is probably adequate for these purposes.

4 Conclusion

There are a wealth of cryptographic options available to anyone seeking tools for secure communications and data storage. Although the options may seem dizzying at first, a few rules of thumb do emerge. For a symmetric cipher, AES is almost always a good choice. For public-key encryption and digital signatures, RSA and El Gamal systems are interchangeable. The particular application and lifetime of the secret dictate the size of the key required.

References

[1] E. Biham. Cryptanalysis of triple modes of operation. *J. Cryptology*, 12(3):161–184, 1999.

[2] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, New York, 1993.

[3] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices Amer. Math. Soc.*, 46(2):203–213, 1999.

[4] D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a small fraction of the private key bits. In *Advances in cryptology—ASIACRYPT'98 (Beijing)*, pages 25–34. Springer, Berlin, 1998.

[5] E. F. Brickell. A survey of hardware implementations of RSA. In G. Brassard, editor, *Proceedings of the Conference on the Theory and Applications of Cryptology held at the University of California, Santa Barbara, California, August 20–24, 1989*, pages xiv+634, New York, 1990. Springer-Verlag.

[6] E. F. Foundation. EFF DES cracker project. available at `http://www.eff.org/DESCracker/`, 1999.

[7] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[8] J. Killian and P. Rogaway. How to protect DES against exhaustive key search (an alalysis of DESX). *J. Cryptology*, 14:17–35, 2001.

[9] K. Kim, S. Park, and S. Lee. Reconstruction of s^2DES s-boxes and their immunity to differential cryptanalysis. In *Proceedings of the 1993 Korea-Japan Workshop on Information Security and Cryptography*, pages 282–291, 1993. Seoul, Korea.

[10] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14:255–293, 2001.

[11] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

[12] M. Matsui. On corrolation between the order of s-boxes and the strength of DES. In *Advances in Cryptology—EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer-Verlag, 1995.

[13] R. C. Merkle and M. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, 1981.

[14] J. Nechvatal et al. Report on the development of the advanced encryption standard (AES). Technical report, National Institute for Standards and Technology, October 2000. available from `http://csrc.nist.gov/encryption/aes/round2/r2report.pdf`.

[15] Pretty good privacy. `http://www.pgp.com`.

[16] B. Schneier. *Applied Cryptography*. John Wiley & Sons, New York, second edition, 1996.

[17] R. D. Silverman. A cost-based security analysis of symmetric and asymmetric key lengths. available from `http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html`, April 2000.

[18] U.S. Department of Commerce/National Institute of Standards and Technology. *Digital Signature Standard (DSS)*, 2002. FIPS PUB 186-2.

[19] P. C. van Oorschot and M. J. Wiener. A known-plaintext attack on two-key triple encryption. In *Advances in Cryptology—EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 318–325. Springer-Verlag, 1991.