An Overview of Public Key Cryptography

Martin E. Hellman

Originally published in
IEEE Communications Magazine
November 1978 — Volume 16, Number 6

Author's Introduction

About 30 years ago when I first started working in cryptography from an unclassified point of view, the climate was far from conducive. On hearing of my new interest, most of my colleagues told me I was crazy to try working in an area where NSA had had a megabudget for decades. "How can you hope to discover anything new. And if you discover anything good, they'll classify it." History has shown that there was more wisdom in their objections than I was willing to allow at the time. But some inner foolishness drew me inexorably to the area. And, as sometimes happens, being foolish turned out to be smart as well. Two other fools, Diffie and Merkle, independently broke with the conventional wisdom. Eventually we found each other and joined forces.

Somehow we knew that cryptography was something the world would need, and expected great commercial demand to materialize within five to 10 years. This estimate turned out to be optimistic, with widespread commercial use taking roughly twice as long to develop. But the scale of deployment has met our grand expectations. Literally millions of people use cryptography daily on the Internet and, as it should be, most do not even know it is protecting them. Their credit card and other sensitive information is transmitted with a high level of cryptographic protection automatically and transparently. Public key cryptography, the subject of this paper, is critical in allowing that ease of use.

Re-reading this article, I am struck by several points:

The decreasing cost of computation has continued to make cryptography ever more ubiquitous. In the 25 years since the article was written, costs have fallen by a further factor of approximately 100,000, so that the US$10 DES chip referred to in the article is today a very small part of a chip or software and has almost no associated cost. While the cryptanalyst's costs have also fallen by this same factor, fortunately his work increases much more rapidly if we do the larger computations that have become economical. The promulgation of the Advanced Encryption Standard (AES) is an important step in this direction.

Another system that appeared after this article should also be mentioned. Schnorr's variant of ElGamal's signature scheme became the basis of the Digital Signature Algorithm.

The need for large safety margins, mentioned toward the end of the article, has been borne out by experience. The exponential work factor (at least in 1978) of what I then called the ax1x2 system has since been cut to a subexponential level. And the subexponential effort to break RSA has also been significantly decreased. The Number Field Sieve is currently the attack of choice on both these systems with large keys.

The system I called the ax1x2 system in this paper has since become known as Diffie-Hellman key exchange. While that system was first described in a paper by Diffie and me, it is a public key distribution system, a concept developed by Merkle, and hence should be called "Diffie-Hellman-Merkle key exchange" if names are to be associated with it. I hope this small pulpit might help in that endeavor to recognize Merkle's equal contribution to the invention of public key cryptography. Space does not permit an explanation of the quirk of fate that seems to have deprived Merkle of the credit he deserves, but a quirk it is.

Some systems will succumb, even with large safety margins, and the Trap Door Knapsack method described in the article was found to be insecure. This emphasizes the experimental nature of cryptography and helps make the point that there is no substitute for certification via mock attack by one's colleagues.

As in 1978, there is still tension between the need for strong cryptography to protect honest people's sensitive information from bad guys and the government's need to spy on the bad guys when they use cryptography to protect their own secrets. The horror of 9/11 helps bring both those needs into focus since cryptography can be used both to protect the planning of terrorist operations and to foil them (e.g., in protecting power networks and other critical infrastructure). The 1996 National Research Council's report, "Cryptography's Role in Securing the Information Society" (CRISIS), explores that theme in depth. The report is notable for its unanimous conclusions even though the participants' backgrounds ranged from the intelligence community to academia.

Lastly, from a personal point of view, as I look back 25 years, I feel a deep sense of gratitude that I was privileged to play a role in such an exciting drama. I thank the IEEE Communications Society for bringing a whiff of nostalgia into my life as I write this retrospective.

Cryptography has been of great importance to the military and diplomatic communities since antiquity but failed, until recently, to attract much commercial attention. Recent commercial interest, by contrast, has been almost explosive due to the rapid computerization of information storage, transmission, and spying.

Telephone lines are vulnerable to wiretapping, and if carried by microwave radio, this need not entail the physical tapping of any wires. The act becomes passive and almost undetectable. It recently came to light that the Russians were using the antenna farms on the roofs of their embassy and consulates to listen in on domestic telephone conversations, and that they had been successful in sorting out some conversations to Congressmen.

Human sorting could be used, but is too expensive because only a small percentage of the traffic is interesting. Instead, the Russians automatically sorted the traffic on the basis of the dialing tones which precede each conversation and specify the number being called. These tones can be demodulated and a microprocessor used to activate a tape recorder whenever an "interesting" telephone number (one stored in memory) is detected. The low cost of such a device makes it possible to economically sort thousands of conversations for even one interesting one.

The problem is compounded in remote computing because the entire "conversation" is in computer readable form. An eavesdropper can then cheaply sort messages not only on the basis of the called number, but also on the content of the message, and record all messages that contain one or more keywords. By including a name or product on this list, an eavesdropper will obtain all messages from, to, or about the "targeted" person or product. While each fact by itself may not be considered sensitive, the compilation of so many facts will often be considered highly confidential.

It is now seen why electronic mail must be cryptographically protected, even though almost no physical mail is given this protection. Confidential physical messages are not written on postcards and, even if they were, could not be scanned at a cost of only $1 for several million words.

This work was supported in part under NSF Grant ENG 10173. At the time of publication Martin Hellman was with the Department of Electrical Engineering at Stanford University.

The Cost of Encryption

Books about World War 11 intelligence operations make it clear that the allies were routinely reading enciphered German messages. The weakness of the Japanese codes was established by the Congressional hearings into the Pearl Harbor disaster, and while it is less well publicized, the Germans had broken the primary American field cipher.

If the major military powers of World War II could not afford secure cryptographic equipment, how is industry to do so in its much more cost-conscious environment?

Encryption is a special form of computation and, just as it was possible to build good, inexpensive, reliable, portable computers in the 1940s, it was impossible to build good (secure), inexpensive, reliable, portable encryption units. The scientific calculator that sells for under $100 today would have cost on the order of a million dollars and required an entire room to house it in 1945.

While embryonic computers were developed during the War (often for code breaking), they were too expensive, unreliable, and bulky for field use. Most computational aids were mechanical in nature and based on gears. Similarly, all of the major field ciphers employed gear-based devices and, just as Babbage's failure indicates the difficulty of building a good computer out of gears, it is also difficult to build a good cryptosystem from gears. The development of general-purpose digital hardware has freed the designers of cryptographic equipment to use the best operations from a cryptographic point of view, without having to worry about extraneous mechanical constraints.

As an illustration of the current low cost of encryption, the recently promulgated national Data Encryption Standard (DES) can be implemented on a single integrated circuit chip, and will sell in the $10 range before long. While some have criticized the standard as not being adequately secure [1], this inadequacy is due to political considerations and is not the fault of insufficient technology.

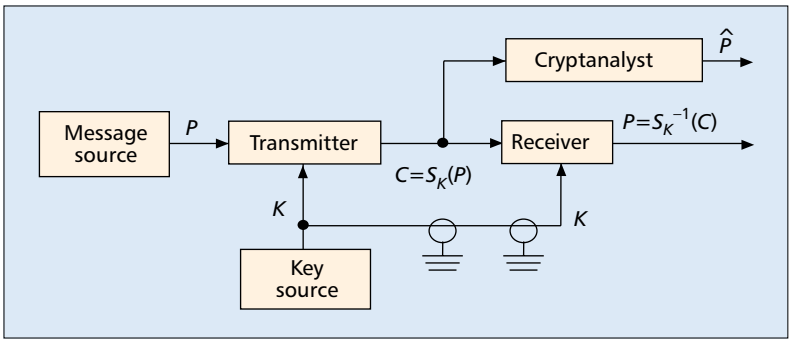


FIGURE 1. *Conventional cryptographic system.*

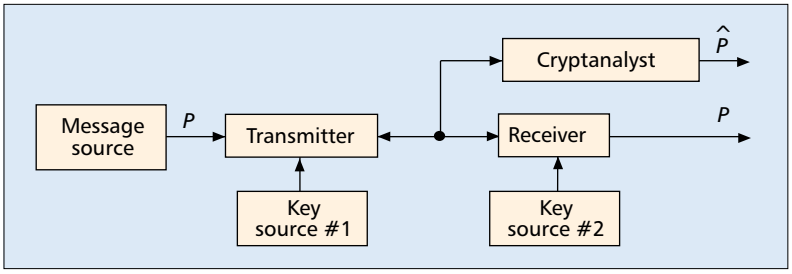


FIGURE 2. *Public key cryptographic system.*

KEY DISTRIBUTION AND PUBLIC KEY SYSTEMS

While digital technology has reduced the cost of encryption to an almost negligible level, there are other major problems involved in securing a communication network. One of the most pressing is key distribution, the problem of securely transmitting keys to the users who need them.

The classical solution to the key distribution problem is indicated in Fig. 1. The key is distributed over a secure channel as indicated by the shielded cable. The secure channel is not used for direct transmission of the plain-text message *P* because it is too slow or expensive.

The military has traditionally used courier service for distributing keys to the sender and receiver. In commercial systems registered mail might be used.. Either way, key distribution is slow, expensive, and a major impediment to secure communication.

Keys could be generated for each possible conversation and distributed to the appropriate users, but the cost would be prohibitive. A system with even a million subscribers would have almost 500 billion possible keys to distribute. In the military, the chain of command limits the number of connections, but even there key distribution has been a major problem. It will be even more acute in commercial systems.

It is possible for each user to have only one key which he shares with the network rather than with any other user, and for the network to use as a master key for distributing conversation-specific keys [2, 3]. This method requires that the portion of the network that distributes the keys (known as the key distribution center or node) be trustworthy and secure.

Diffie and Hellman [4] and independently Merkle [5] have proposed a radically different approach to the key distribution problem. As indicated in Fig. 2, secure communication takes place without any prearrangement between the conversants and without access to a secure key distribution channel. As indicated in the figure, two-way communication is allowed and there are independent random number generators at both the transmitter and the receiver. Two-way communication is essential to distinguish the receiver from the eavesdropper. Having random number generators at both ends is not as basic a requirement, and is only needed in some implementations.

The situation is analogous to having a room full of people who have never met before and who are of equal mathematical ability. I choose one other person in the room and, with everyone else listening, give him instructions that allow the two of us to carry on a conversation that no one else can understand. I then choose another person and do the same with him.

This sounds somewhat impossible and, from one point of view, it is. If the cryptanalyst had unlimited computer time he could understand everything we said. But that is also true of most conventional cryptographic systems — the cryptanalyst can try all keys until he finds the one that yields a meaningful decipherment of the intercepted message. The real question is whether we can, with very limited computations, exchange a message that would take the cryptanalyst eons to understand using the most powerful computers envisionable.

A public key cryptosystem [4] has two keys, one for enciphering and one for deciphering. While the two keys effect inverse operations and are therefore related, there must be no easily computed method of deriving the deciphering key from the enciphering key. The enciphering key can then be made public without compromising the deciphering key so that anyone can encipher messages, but only the intended recipient can decipher messages.

The conventional cryptosystem of Fig. 1 can be likened to a mathematical strongbox with a resettable combination lock. The sender and the receiver use a secure channel to agree on a combination (key) and can then easily lock and unlock (encipher and decipher) messages, but no one else can.

A public key cryptosystem can be likened to a mathematical strongbox with a new kind of resettable combination lock that has two combinations, one for locking and one for unlocking the lock. (The lock does not lock if merely closed.) By making the locking combination (enciphering the key) public anyone can lock up information, but only the intended recipient who knows the unlocking combination (deciphering key) can unlock the box to recover the information.

Public key and related cryptosystems have been proposed by Merkle [5], Diffie and Hellman [4], Rivest *et al*. [6], Merkle and Hellman [7], and McEliece [8]. We will only outline the approaches, and the reader is referred to the original papers for details.

The RSA (Rivest *et al*.) scheme [6] is based on the fact that it is easy to generate two large primes and multiply them together, but it is much more difficult to factor the result. (Try factoring 518940557 by hand. The try multiplying 15107 by 34351.) The product can therefore be made public as part of the enciphering key without compromising the factors which effectively constitute the

deciphering key. By making each of the factors 100 digits long, the multiplication can be done in a fraction of a second, but factoring would require billions of years using the best known algorithm.

As with all public key cryptosystems there must be an easily implemented algorithm for choosing an enciphering-deciphering key pair, so that any user can generate a pair, regardless of his mathematical abilities. In the RSA scheme the key generation algorithm first selects two large prime numbers p and q and multiplies them to produce $n = pq$. The Euler's function is computed as $\phi(n) = (p-1)(q-1)$. ($\phi(n)$ is the number of integers between 1 and n that have no common factor with n. Every p^{th} number has p as a common factor with n and every q^{th} number has q as a common factor with n.) Note that it is easy to compute $\phi(n)$ if the factorization of n is known, but computing $\phi(n)$ directly from n is equivalent in difficulty to factoring n [6].

$\phi(n)$ as given above has the interesting property that for any integer a between 0 and $n-1$ (the integers modulo n) and any integer k

$$a^{k\phi(n)+1} = a \cdot \bmod n. \tag{1}$$

Therefore, while all other arithmetic is done modulo n, arithmetic in the exponent is done modulo $\phi(n)$.

A random number E is then chosen between 3 and $\phi(n)-1$ and which has no common factors with $\phi(n)$. This then allows

$$D = E^{-1} \bmod \phi(n) \tag{2}$$

to be calculated easily using an extended version of Euclid's algorithm for computing the greatest common divisor of two numbers [9, p. 315, problem 15; p. 523, solution to problem 15].

The information (E,n) is made public as the enciphering key and is used to transform unenciphered, plain text messages into ciphertext messages as follows: a message is first represented as a sequence of integers each between 0 and $n-1$. Let P denote such integer. Then the corresponding ciphertext integer is given by the relation

$$C = P^E \bmod n. \tag{3}$$

The information (D,n) is used as the deciphering key to recover the plain text from the ciphertext via

$$P = C^D \bmod n. \tag{4}$$

These are inverse transformations because from [3], [2], and [1]

$$C^D = P^{ED} = \mathrm{P}^{k\phi(n)+1} = P \tag{5}$$

As shown by Rivest *et al.*, computing the secret deciphering key from the public enciphering key is equivalent in difficulty to factoring n.

As a small example, suppose $p = 5$ and $q = 11$. Then $n = 55$ and $\phi(n) = 40$. If $E = 7$ then $D = 23$ ($7 \times 23 = 161 = 1 \bmod 40$). If $P = 2$ then

$$C = 2^7 \bmod 55 = 18 \tag{6}$$

and

$$C^D = 18^{23} \bmod 55 \tag{7}$$

$$= 18^1 18^2 18^4 18^{16} \tag{8}$$

$$= 18\ 49\ 36\ 26 \mod 55 \tag{9}$$

$$= 2 \tag{10}$$

which is the original plain text.

THE RIVEST-SHAMIR-ADLEMAN PUBLIC KEY SCHEME

Design

Find two large prime numbers p and q, each about 100 decimal digits long. Let $n = pq$ and $\psi = (p-1)(q-1)$.

Choose a random integer E between 3 and ψ that has no common factors with ψ. Then it is easy to find an integer D that is the "inverse" of E modulo ψ, that is, $D \cdot E$ differs from 1 by a multiple of ψ.

The public information consists of E and n. All other quantities here are kept secret.

Encryption

Given a plain text message P that is an integer between 0 and $n-1$ and the public encryption number E, form the ciphertext integer

$C = P^E \bmod n$.

In other words, raise P to the power E, divide the result by n, and let C be the remainder. (A practical way to do this computation is given in the text of Hellman's paper.)

Decryption

Using the secret decryption number D, find the plain text P by

$P = C^D \bmod n$.

Cryptanalysis

In order to determine the secret decryption key D, the cryptanalyst must factor the 200 or so digit number n. This task would take a million years with the best algorithm known today, assuming a 1μs instruction time.

Note that enciphering and deciphering each involve an exponentiation in modular arithmetic and that this can be accomplished with at most $2(\log_2 n)$ multiplication mod n. As indicated in [8], to evaluate $Y = a^X$, the exponent X is represented in binary form, the base a is raised to the 1st, 2nd, 4th, 8th, etc. powers (each step involving only one squaring or multiplication), and the appropriate set of these are multiplied together to form Y.

Merkle and Hellman's method [7] makes use of trapdoor knapsack problems. The knapsack problem is a combinatorial problem in which one is given a vector of n integers, $\boldsymbol{a}$, and an integer S which is a sum of a subset of the $\{a_i\}$. The problem is to solve for the subset, or equivalently, for the binary vector $\boldsymbol{x}$ which is the solution to the equation

$$S = \boldsymbol{a} * \boldsymbol{x}. \tag{11}$$

While the knapsack problem is very difficult to solve in general, there are specific cases that are easy to solve. For example, if the knapsack vector is

$$\boldsymbol{a}' = (171, 197, 459, 1191, 2410) \tag{12}$$

then given any S', $\boldsymbol{x}$ is easily found because each component of $\boldsymbol{a}'$ is larger than the sum of the preceding components. If $S'=3798$, then it is seen that x_5 must be 1 because, if it were 0, $a_5' = 2410$ would not be in the sum and the remaining elements sum to less than S'. After subtracting the effect of a_5' from S', the solution continues recursively and establishes that $x_4 = 1$, $x_3 = 0$, $x_2 = 1$, and $x_1 = 0$.

The knapsack vector

$$\boldsymbol{a} = (5457, 4213, 5316, 6013, 7439) \tag{13}$$

does not possess the property that each element is larger than the sum of the preceding components, and the simple method of solution is not possible. Given $S = 17665$, there is no obvious

A true digital signature must be a number (so it can be sent in electronic form) that is easily recognized by the receiver as validating the particular message received, but which could only have been generated by the sender.

method for finding that $\boldsymbol{x} = (0,1,0,1,1)$ other than trying almost all 2^5 subsets.

But it "just so happens" that if each component of $\boldsymbol{a}$ is multiplied by 3950 modulo 8443 the vector $\boldsymbol{a}'$ of [12] is obtained. By performing the same transformation on S, the quantity $S' = 3798$ is obtained. It is now seen that there is a simple method for solving for $\boldsymbol{x}$ in the equation

$$S = a^{*}\boldsymbol{x} \tag{14}$$

by transforming to the easily solved knapsack problem

$$S' = \boldsymbol{a}' * \boldsymbol{x}. \tag{15}$$

The two solution $\boldsymbol{x}$ are the same provided the modulus is greater than the sum of the $\{a_i'\}$.

The variables of the transformation (the multiplier 3950 and the modulus 8443) are secret, trap-door information used in the construction of the trap-door knapsack vector $\boldsymbol{a}$. There is no apparent easy way to solve knapsack problems involving $\boldsymbol{a}$ unless one knows the trap-door information.

When $\boldsymbol{a}$ is made public, anyone can represent a message as a sequence of binary $\boldsymbol{x}$ vectors and transmit the information securely in the corresponding sums, $S = \boldsymbol{a} * \boldsymbol{x}$. The intended recipient uses his trap-door information (secret deciphering key) to easily solve for $\boldsymbol{x}$, but no one else can do this. Of course the $\boldsymbol{a}$ vector must be significantly longer than that used in this small, illustrative example.

McEliece's public key cryptosystem [8] is based on algebraic coding theory. Goppa codes are highly efficient error correcting codes [10], but their ease of error correction is destroyed if the bits that make up a codeword are scrambled prior to transmission. To generate a public enciphering key, a user first selects a Goppa code chosen at random from a large set of possible codes. He then selects a permutation of the codeword bits, computes the generator matrix associated with the scrambled Goppa code, and makes it public as his enciphering key. His secret deciphering key is the permutation and choice of Goppa code.

Anyone can easily encode information (scrambling does not greatly increase the difficulty of encoding since the scrambled code is still linear), add a randomly generated error vector, and transmit this. But only the intended recipient knows the inverse permutation that allows the errors to be corrected easily.

McEliece estimates that a block length of 1000 bits with 500 information bits should foil cryptoanalysis using the best currently known attacks.

The other two known methods for communicating securely over an insecure channel without securely transmitting a key are not true public key cryptosystems. Rather, they are public key distribution systems that are used to securely exchange a key over an insecure channel without any prearrangement, and that key is then used in a conventional cryptosystem.

Merkles's technique [5] involves an exchange of "puzzles." The first user generates n potential keys and hides them as the solution to n different puzzles, each of which costs n units to solve. The second user chooses one of the n puzzles at random, solves it, and sends a test message encrypted in the associated key. The first user determines which key was chosen by trying all n of them on the test message.

The cost to the first user is proportional to n. He must generate and store n keys, generate and transmit n puzzles, and try n keys on the test message. The cost to the second user is also proportional to n because he must solve one puzzle that was designed to have solution cost equal to n.

The cost to an eavesdropper appears to grow as n^2. He can try solving puzzles at random and see if the associated key (solution) agrees with the test message. On the average, he must solve $n/2$ puzzles, each at a cost of n.

Diffie and Hellman [4] describe a public key distribution system based on the discrete exponential and logarithm functions. If q is a prime number and a is a primitive element, then X and Y are in a 1:1 correspondence for $1 \leq X, Y \leq (q-1)$ where

$$Y = a^{x} \mod q \tag{16}$$

and

$$X = \log_a Y \quad \text{over GF}(q). \tag{17}$$

While the discrete exponential function (16) is easily evaluated, as in [7] and [8], no general, fast algorithms are known for evaluating the discrete logarithm function (17). Each user chooses random element X and makes the associated Y public. When users i and j wish to establish a key for communicating privately they use

$$K_{ij} = a^{X_i X_j} \tag{18}$$

$$= (Y_i)^{X_j} = (Y_j)^{X_i}. \tag{19}$$

Equation 19 demonstrates how both users i and j use the easily computed discrete exponential function to calculate K_{ij} from their private and the other's public information. An opponent who knows neither user's secret information can compute K_{ij} if he is willing to compute a discrete logarithm, but that can be made computationally infeasible using the best currently known algorithms [11].

The various public key systems are compared in a later section.

Digital Signatures

Business runs on signatures, and until electronic communications can provide an equivalent of the written signature, it cannot fully replace the physical transportation of documents, letters, contracts, etc.

Current digital authenticators are letter or number sequences that are appended to the end of a message as a crude form of signature. By encrypting the message and authenticator with a conventional cryptographic system, the authenticator can be hidden from prying eyes. It therefore prevents third-party forgeries. But because the authentication information is *shared* by the sender and receiver, it cannot settle disputes as to what message, if any, was sent. The receiver can give the authentication information to a friend and ask him to send a signed message of the receiver's choosing. The legitimate sender of messages will of course deny having sent this message, but there is no way to tell whether the sender or receiver is lying. The whole concept of a contract is embedded in the possibility of such disputes, so stronger protection is needed.

A true digital signature must be a number (so it can be sent in electronic form) that is easily recognized by the receiver as validating the particular message received, but which could only have been generated by the sender. It may seem impossible for the receiver to be able to recognize a number that he cannot generate, but such is not the case.

While there are other ways to obtain digital signatures, the easiest to understand makes use of the public key cryptosystems discussed in the last section. The ith user has a public key E_i and a secret key D_i. This notation was chosen because E_i was used to encipher and D_i was used to decipher. Suppose, as in the RSA scheme, the enciphering function is **onto**, that is, for every integer C less than n, there exists an integer m for which $E_i(m) = C$. Then we can interchange the order of operations and use D_i first to sign the message and E_i second to validate the signature. When user i wants to sign and send a message M to user j, he operates on M with his secret key D_i to obtain

$$C = D_i(M) \tag{20}$$

which he then sends to user j. User j obtains i's public key E_i from a public file and operates with it on C to obtain M

$$E_i(C) = E_i[D_i(M)] = M \tag{21}$$

User j saves C as proof that message M was sent to him by user i. No one else could have generated C, because only i knows D_i. And if j tries to change even one bit in C, he changes its entire meaning (such error propagation is necessary in a good cryptosystem).

If i later disclaims having sent message M to user j, then j takes C to a "judge" who accesses the public file and checks whether $E_i(C)$ is a meaningful message with the appropriate date, time, address, name, etc. If it is, the judge rules in favor of j. If it is not, the ruling is in favor of i.

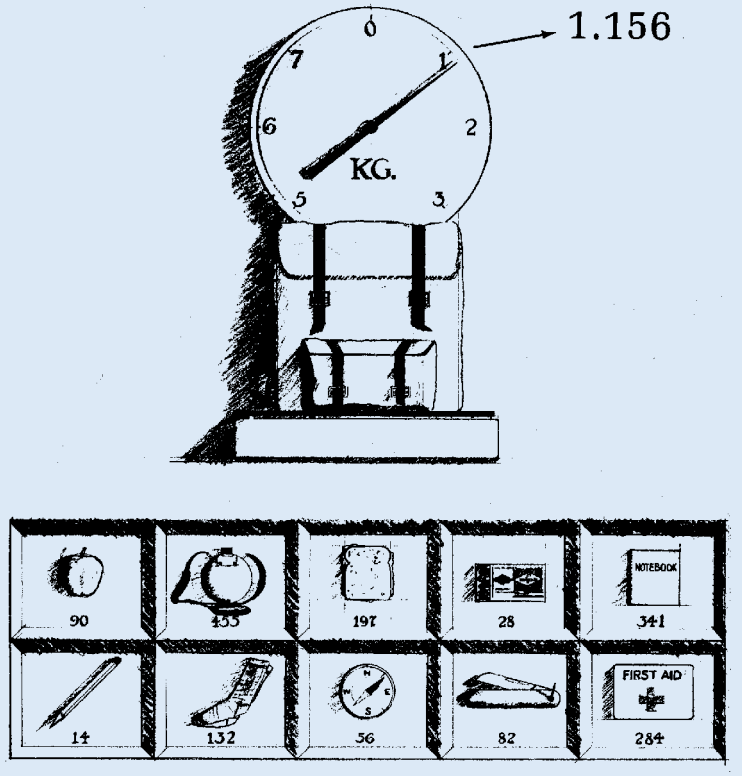
Digital signatures have an advantage over written signatures because written signatures look the same, independent of the message. My signature is supposed to look the same on a $100 check as on a $1000 check, so a dishonest recipient can try to alter the check. Similarly, if a photostat of a contract is acceptable as proof, a dishonest person can alter the contract and make a copy that hides the alteration. Such mischief is impossible with digital signatures, provided the signature system is truly secure.

The disadvantage of digital signatures is that the ability to sign is equivalent to possession of a secret key. This key will probably be stored on a magnetic card which, unlike the ability to sign one's name, can be stolen.

Comparison of Public Key Systems

This section compares the public key systems that have been proposed. Speed, ease of signature generation, and certain other characteristics can be compared more readily than the all important question of security level. We can compare the security level using the best known methods for breaking each system, but there is the danger that better methods will be found that will change the relative rankings.

The Knapsack Problem



The knapsack is filled with a subset of the items shown, with weights indicated in grams. Given the weight of the filled knapsack, 1156 grams, can you determine which of the items are contained in the knapsack? (The scale is calibrated to deduct the weight of the empty knapsack.)

This simple version of the classic knapsack problem generally becomes computationally feasible when there are 100 items rather than 10 as in this example. However, if the set of weights for the items happens to have some nice properties known only to someone with special "trap-door" information, then that person can quickly decipher the secret information, i.e., a 100-bit binary word that specifies which of the items are in the knapsack.

If signatures are desired, attention should be directed to the RSA [6] and trap-door knapsack systems [7]. The RSA scheme yields signatures directly. While the trap-door knapsack signature method described in [7] is not direct, Merkle and Reeds have developed a method for generating "high-density" trap-door knapsacks that simplify signature generation, and Shamir has recently suggested a direct method for obtaining signatures. Both of these approaches are not yet published.

The $a(X_1X_2)$ and Goppa code methods do not appear to lend themselves to signatures, but Mekle has developed a puzzle-like technique for generating signatures.

So far, as storage requirements for the public file, the $a(X_1X_2)$ and RSA schemes are most interesting. Each requires on the order of 500 bits of storage per user. The trap-door knapsack scheme requires on the order of 100 kbits of storage per user, and the Goppa code method requires on the order of a megabit per user. Merkle's puzzle scheme is not really suited to public file storage and rather depends on transmission of public information at the start of each

As computers become faster and more parallel, the time for cryptanalysis also falls. A 1ns computer with millionfold parallelism might reduce the time estimates given in the tables by a factor of 109.

b (bits)	100	200	300	500	750	1000
Operations	1.1×10^{15}	1.3×10^{30}	1.4×10^{45}	1.8×10^{75}	7.7×10^{112}	3.3×10^{150}
Time (years)	36	4×10^{16}	5×10^{31}	6×10^{61}	2×10^{99}	1×10^{137}

Table 1.

b (bits)	100	200	300	500	750	1000
Operations	2.8×10^{7}	2.3×10^{11}	2.9×10^{14}	3.6×10^{19}	5.8×10^{24}	1.8×10^{29}
Time	30s	3 days	9 years	1 Myr	2 Gyr	6×10^{15}yrs

Table 2.

new conversation. The transmitted information must be on the order of a gigabit before significant levels of security are afforded.

Instead of storing each user's public key in a public file (similar to a telephone book), Kohnfelder [12] has suggested having the system give each user a signed message, or certificate, statting that user's public key. The certificate could be stored by the user on a magnetic card, and transmitted at the start of a conversation. This method converts public file storage requirements into transmission requirements. The system's public key would be needed to check the certificate and could be published widely. Protecting the system's secret key might be easy because no one else ever has to use it and it could be destroyed after it was used to certify a group of users.

Computation time on the part of the legitimate users is smallest with the trap-door knapsack method. The $a^{(X_1X_2)}$ and RSA schemes each require several hundred times as much computation, but are still wihin reason. Merkle's technique requires even more computation. The Goppa code technique is extremely fast for enciphering, requiring approximately 500 XORs on 1000-bit vectors, but I have not yet estimated its deciphering requirements.

Turning to security level, Merkle's puzzle method [5] has the advantage of being the most solid method for communicating securely over an insecure channel. That is, it is extremely doubtful that a better method will be found for breaking it. Unfortunately, it is also the least secure, using the best known algorithm. Its work factor (ratio of cryptanalytic effort to enciphering and deciphering effort, using the best known algorithms) is only n^2:n. Since encryption should cost on the order of \$0.01 and cryptanalysis should cost on the order of @10 million or more, this ratio needs to be 10^9 or more and corresponds to $n = 10^9$. If all of the enciphering and deciphering effort were in computation, this might be possible in the near future (a \$10 microprocessor can execute on the order of 1 million instructions per second), but Merkle's method requires n transmissions as well as n operations on the part of the legitimate users. Current technology therefore limits Merkle's scheme to $n \approx 10{,}000$ which corresponds to approximately 500 kbits of transmission. If fiber optic or other low cost, ultra-high-bandwidth communication links become available, Merkle's technique would become of greater practical interest.

Diffie and Hellman's exponentiation method [4] requires the legitimate users to perform an exponentiation in modular arithmetic while the best known cryptanalytic method requires the computation of a logarithm in modular arithmetic. Exponentiation is easily accomplished in at most $2b$ multiplications, much as in [8], where b is the number of bits in the representation of the modulus. Each multiplication can be accomplished with at most $2b$ additions or subtractions, and each of these operations involves at most b gate delays for the propagation of carry signals. Overall, an exponentiation in modular arithmetic can be accomplished in at most $4b^3$ gate delays.

Computation of a logarithm in modular arithmetic is much more complex, and the best currently known algorithm [11] requires $2^{b/2}$ or more operations provided the modulus is properly chosen. Each operation involves a multiplication, or $2b^3$ gate delays. The work factor is therefore exponential in b.

If $b = 500$, the 500 million gate delays are required at the legitimate users' terminals. With current technology this can be accomplished in several seconds, a not unreasonable delay for establishing a key during initial connection. Using $b = 500$ results in the cryptanalyst having to do more than 10^{75} times as much work as the legitimate users, a very safe margin. The real question is whether better methods exist for computing logarithms in modular arithmetic, or if it is even necessary to compute such a logarithm to break this system.

Table 1 gives the number of operations and time required for cryptanalysis for various values of b assuming a 1μs instruction time.

The storage requirements of this system are small. The public file stores a single b-bit number for each user and only several b-bit words of memory are required at the transmitter and receiver, so that single-chip implementation is possible for b on the order of 500.

The RSA system [6] also requires that the legitimate users perform a modular exponentiation, but cryptanalysis is equivalent to factoring a b-bit number. Schroeppel has developed a new, as yet unpublished factoring algorithm that appears to require approximately $\exp\{[\ln(n)\ln(\ln n))]^{1/2}\}$ machine cycles where $n = 2^b$ is the

number to be factored. Table 2 gives the number of operations and time to factor a b-bit number again assuming a 1μs instruction time.

Public file storage for the RSA scheme is reasonable, being several hundred to a thousand bits per user. Memory requirements at the transmitter and receiver are also comparable to the $a(X_1X_2)$ scheme, so that a single-chip device can be built for enciphering and deciphering.

The best known method of crytpanalyzing the trap-door knapsack system requires on the order of $2^{n/2}$ operations where n is the size of the knapsack vector. Enciphering requires at most n additions, so the work factor is exponential. If n is replaced by b, Table 1 gives the cryptanalytic effort required for various values of n, so $n \geq 200$ provides relatively high security levels. Since each element of the $\boldsymbol{a}$ vector is approximately $2n$ bits long, if $n = 200$, the public storage is approximately 80 kbits/user. Memory requirements at the transmitter and receiver are on the same order.

Both enciphering and deciphering require less computation than either the $a(X_1X_2)$ or RSA scheme. Enciphering required at most n additions and deciphering requires one multiplication in modular arithmetic, followed by at most n subtractions.

Care must be exercised in interpreting these tables. First, they assume that the cryptanalyst uses the best currently known method, and there may be much faster approaches. For example, prior to the development of Schroeppel's algorithm, the best factoring algorithm appeared to require $\exp\{[2\ln(n)\ \ln(\ln n)]^{1/2}\}$ operations. When $b = 200$, that would have predicted that 360 years, not three days, would be required for cryptanalysis. There is the danger that even faster algorithms will be found, necessitating a safety margin in our estimates. As demonstrated by this example, the safety margin is needed in the exponent, not the mantissa.

A similar comment applies to the seemingly higher security level afforded by the $a(X_1X_2)$ and trap-door knapsack methods when compared to the RSA scheme. For a given value of b the two tables show that the RSA scheme requires much less computation to break, using the best currently known techniques. But it is not clear whether this is because the factoring is inherently easier than computing discrete algorithms or solving knapsack problems, or whether it is due to the greater study that has been devoted to factoring.

As computers become faster and more parallel, the time for cryptanalysis also falls. A 1ns computer with millionfold parallelism might reduce the time estimates given in the tables by a factor of 10^9.

Conclusions

We are in the midst of a communications revolution that will impact many aspects of people's every day lives. Cryptography is an essential ingredient in this revolution, and is necessary to preserve privacy from computerized censors capable of scanning millions of pages of documents for even one sensitive datum. The public key and digital signature concepts are necessary in commercial systems because of the large number of interconnections that are possible, and because of the need to settle disputes.

A major problem that confronts cryptography is the certification of these systems. How can we decide which proposed systems really are secure, and which only appear to be secure? Proofs are not possible using the currently developed theory of computational complexity and, while such proofs may be possible in the future, something must be done immediately. The currently accepted technique for certifying a cryptographic system as secure is to subject it to a mock attack under circumstances that are extremely favorable to the cryptanalyst and unfavorable to the system. If the system resists such a concerted attack under unfavorable conditions, it is hoped that it will also resist attacks by one's opponents under more realistic conditions.

Governments have built up expertise in the certification area, but due to security constraints, this is not currently available for certification of commercially oriented systems. Rather, this expertise in the hands of a foreign government poses a distinct threat to a nation's businesses. It has even been suggested that poor or nonexistent encryption will lead to international economic warfare, a concern of importance to national security. (There is speculation that this occurred with the large Russian grain purchases of several years ago.)

There is a tradeoff between this and other national security considerations that needs to be resolved, but the handling of the national data encryption standard indicates that public discussion and resolution of the tradeoff is unlikely unless individuals make their concern known at a technical and political level.

A major problem that confronts cryptography is the certification of these systems. How can we decide which proposed systems really are secure, and which only appear to be secure?

References

[1] W. Diffie and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *Computer*, pp. 74–84, June 1977.

[2] D. Branstad, "Encryption Protection in Computer Data Communications," presented at the *IEEE Fourth Data Communications Symposium*, Oct. 7–9, 1975, Quebec, Canada.

[3] *IBM Syst. J., (Special Issue on Cryptography)*, vol. 17, no. 2, 1978.

[4] W. Diffie and M. E. Hellman, " New Directions in Cryptography," *IEEE Trans. Info. Theory*, vol. IT-22, Nov. 1976, pp. 644–54.

[5] R. C. Merkle, "Secure Communication over an Insecure Channel," *Commun. Ass. Comp. Mach.*, vol. 21, Apr. 1978, pp. 294–99.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "On Digital Signatures and Public Key Cryptosystems," *Commun. Ass. Comp. Mach.*, vol. 21, Feb. 1978, pp. 120–26.

[7] R. C. Merkle and M. E. Hellman, "Hiding Information and Signatures in Trap-door Knapsacks," *IEEE Trans. Info. Theory*, vol. IT-24, Sept. 1978, pp 525–30.

[8] R. J. McEliece, "A Public Key System based on Algebraic Coding Theory," JPL DSN Progress Rep., 1978.

[9] D. E. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Reading, MA: Addison-Wesley, 1969.

[10] E. R. Berlekamp, "Goppa Codes," *IEEE Trans. Info. Theory*, vol. IT-19, Sept. 1973, pp. 590–92.

[11] S. C. Pohlig and M. E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," *IEEE Trans. Info. Theory*, vol. IT-24, Jan. 1978, pp. 106–10.

[12] L. Kohnfelder, *Towards a Practical Public-Key Cryptosystem*, MIT Lab. For Comput. Sci., June 1978.

Learn how public key cryptography works as well as common uses for this cryptographic method. A Definition of Public Key Cryptography. Sometimes referred to as asymmetric cryptography, public key cryptography is a class of cryptographic protocols based on algorithms. This method of cryptography requires two separate keys, one that is private or secret, and one that is public. Public key cryptography uses a pair of keys to encrypt and decrypt data to protect it against unauthorized access or use. Network users receive a public and private key pair from certification authorities. If other users We briefly mentioned Asymmetric Ciphers earlier in this book. In this and following chapters we will describe how they work in much more detail. The discovery of public key cryptography revolutionized the practice of cryptography in the 1970s. In public key cryptography, the key used to encrypt a message is not the same as the key used to decrypt it. This requires an asymmetric key algorithm. Public Key Cryptography (PKC): Uses one key for encryption and another for decryption; also called asymmetric encryption. Primarily used for authentication, non-repudiation, and key exchange. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information, providing a digital fingerprint. Primarily used for message integrity. FIGURE 1: Three types of cryptography: secret key, public key, and hash function. 3.1. Secret Key Cryptography. Secret key cryptography methods employ a single key for both encryption and decryption. As shown in Figure 1A, the sender use Public key cryptography with digital signatures: A digital signature with public-key cryptography securing a message is created in the following way. First, the message is digitally signed like explained above. Then, this bundle is encrypted with the senderâ€™s private key, and again with the receiverâ€™s public key. Looking at it like hypothetical function calls, it may look something like this: public_key_of_recipient(private_key(message_hashing(message) + message + type of hashing algorithm)). Keep this a secret. Decrypting secured messages with digital signatures: When the recipient receives t