MIT/LCS/TM-107

A FAST SIGNATURE SCHEME

Adi Shamir

May 1978

MIT/LCS/TM-107

A FAST SIGNATURE SCHEME

Adi Shamir

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

A Fast Signature Scheme

by

Adi Shamir

Department of Mathematics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

May, 1978

Abstract: In this paper we propose a new scheme for generating and verifying "electronic signatures" in public-key communications. The scheme is based on the difficulty of solving the knapsack problem, and its two main advantages over previous schemes are speed and simplicity.

Key words: Electronic Signatures, public-key communications, knapsack problem, knapsack system; coding techniques

This research was supported by the Office of Naval Research grant number N00014-76-C-0366.

1. Introduction

When two parties with conflicting interests (such as a bank and a customer, or two competing companies) are communicating, it is essential that the originator of every message (party A) sign it, and that the receiver of every message (party B) check the signature. This should give the parties two kinds of protection:

(i) Both party A and party B should be protected against forged messages, planted in the communication channel by a third party C which pretends to be party A.

(ii) Party A should be protected against messages forged by party B, which claims to have received them (properly signed) from party A.

The first kind of protection can be guaranteed by using appropriate coding techniques, which are known only to A and B. The second kind of protection seems to be harder to obtain, since B should know enough about the way A signs its messages in order to recognize them, and yet should be unable to generate them. Note that when the signature is electronic (e.g., a certain pattern of 0's and 1's), it must be message dependent — otherwise B can copy A's signature and attach it to any other message.

If the network of communicating parties is sufficiently big (e.g., the network of phone or mail users), it is completely impractical to use a distinct and secret signature algorithm for every pair of potential users. In their excellent paper [1],

Diffie and Hellman introduce the notion of a "public key cryptosystem", in which (among other things) each user makes public a quick method for recognizing his signatures. The resultant "signature directory" is available to anyone, and thus two participating parties can start sending signed messages without any special setup (such as the exchange of secret keys via special couriers). In the context of a public-key cryptosystem, protection problem (i) becomes a variant of protection problem (ii), since A and B cannot share any information which is kept secret from C.

Three main solutions have been suggested so far for the electronic signature problem. The first one (chronologically) is due to Rabin [2], and it is based on probabilistic ideas. Its main drawback, however, is a fairly complicated signing and verification procedure. The second one is the Rivest-Shamir-Adleman cryptographic system [3], which solves both the signature and the security problem in public-key communications. The main problem with this system is that it is relatively slow, since messages are signed by performing hundreds of high-precision multiplications. Finally, the trapdoor knapsacks developed by Merkle and Hellman [4] to encode data in public-key cryptosystems can be used to generate some signatures, but the system is quite awkward to use since only a tiny fraction of the set of all messages is signable.

The purpose of this paper is to propose a new signature scheme in which the emphasis is on speed and simplicity — the

main deficiencies in previous systems. Both the signing and the verification can be done by performing just additions and subtractions — there are no high-precision multiplications or complicated bit operations involved. The electronic implementation of the scheme can be simple and compact, and it is particularly useful in high-speed applications (e.g., keeping rapidly changing computerized information signed at all times).

This paper describes only the basic signature scheme. There are many possible variations, improvements and additions to the scheme, which might increase its security or simplify its implementation. We do not have any proof that the scheme is (or can be made) "unbreakable", but this is the case with most cryptographic systems (including those mentioned above). The only known method to certify its security is to expose it to a concentrated but unsuccessful cryptanalytic attack; the reader is urged to participate in this effort by trying to break the proposed scheme, and to find its variants which withstand the cryptanalytic attack best. One line of attack which deserves special attention and close study is the statistical method mentioned in section 3.1.

2. The basic scheme

2.1 Knapsack systems

The knapsack problem considered in this paper is an extension of the one defined in Karp [5]:

Given $k+2$ integers $a_1,\ldots,a_k,n$ and m, find a solution $c_1,\ldots,c_k$ (if one exists) for the modular equation

$$(1) \qquad m = \sum_{j=1}^{k} c_j a_j \pmod{n}$$

in which each c_j is a small integer in the range $0 \leq c_j \leq \log n$.

It is easy to extend Karp's original reduction (from the exact covering problem) to show that this variant is also an NP-complete problem, and thus the worst case complexity of any algorithm which solves it is strongly believed to be non-polynomial.

The signature schemes we develop are based on particular instances of this problem. We use the words a knapsack system to denote the knapsack problem in which $a_1,\ldots,a_k$ and n are fixed numbers, and the only variable is m. The interesting values of m are in the interval $0 \leq m < n$, and thus any knapsack system is just a finite collection of instances of the knapsack problem. Since (at least in theory) it is possible to extract the solutions of these instances from a finite precomputed table, it is hard to define the difficulty of solving a particular knapsack system in a precise mathematical way. Our usage of "easy" and "difficult" in this paper will thus be based on their intuitive meaning: A knapsack system is "difficult" if the only apparent way of solving its instances is by a (more or less) exhaustive search, and "easy"

if substantially shorter methods exist.

2.2. Knapsack systems and signatures.

A knapsack system can be used to generate signatures in the following way. Party A chooses and publishes a knapsack system $a_1,\ldots,a_k,n$ which is apparently difficult, but which can actually be solved quickly by using some secret structure embedded in it. Given a message m, A signs it by using his shortcut method to find a solution for equation (1), and sends the k-tuple $(c_1,\ldots,c_k)$ as his signature, along with m. The receiver B can easily plug the message m, the signature $c_1,\ldots,c_k$ and the published numbers $a_1,\ldots,a_k$ and n into (1), and verify that the equation holds. If party C (or B himself) wants to forge A's signature on another message m', it has to solve that particular instance of A's knapsack system. Note that in order to be useful in generating signatures, A's knapsack system must be generative, i.e., all its instances (with $0 \leq m < n$) must have at least one solution.

2.3. How to construct knapsack systems.

In order to make the construction process clearer, we use typical numbers and sizes in the description that follows. All the numbers involved are 100 bits long, and the knapsack system contains 200 numbers $a_1,\ldots,a_{200}$.

Party A starts by choosing a random 100 bit prime number n and a 100×200 0-1 matrix $[e_{ij}]$ whose entries are chosen at random. The numbers $a_1,\ldots,a_{200}$ are defined to be some solution of the

following system of modular linear equations (we use row indexes 0 to 99 in order to simplify the subsequent formulas):

$$(2)\qquad \begin{bmatrix} e_{0,1} & \cdots & e_{0,200} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ e_{99,1} & \cdots & e_{99,200} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ a_{200} \end{bmatrix} = \begin{bmatrix} 2^0 \\ 2^1 \\ \cdot \\ \cdot \\ 2^{99} \end{bmatrix} \pmod n$$

Since there are only one hundred equations in two hundred unknowns, we can choose $a_{101},\ldots,a_{200}$ at random and solve for $a_1,\ldots,a_{100}$. The probability of getting stuck in this process is very small, since the reduced 100×100 matrix (which is the left half of $[e_{ij}]$) can be singular only if its determinant is an exact multiple of the huge prime n. Even if this happens, we just have to choose another random matrix and try again.

The generated numbers a_j are randomly-looking 100-bit integers, which have the property that any power of 2 between 2^0 and 2^{99} can be expressed as the sum of some subset of them: $2^i = \sum_j e_{ij}a_j$. However, the problem of reversing the process and finding the coefficients e_{ij} with which each power 2^i can be represented (merely by looking at the a_j's) is essentially the knapsack problem, and thus is assumed to be extremely hard.

2.4. How to sign — preliminary approach.

In order to sign a given message m, A writes it as a sum of powers of 2, $m = \sum_{i=0}^{99} m_i 2^i$, where m_i is the i^{th} bit in m's binary representation. Each power of 2 can be represented

in terms of the a_j's, and thus

$$(3) \qquad m = \sum_{i=0}^{99} m_i 2^i = \sum_{i=0}^{99} m_i \left(\sum_{j=1}^{200} e_{ij} a_j \right)$$

$$= \sum_{j=1}^{200} \left(\sum_{i=0}^{99} m_i e_{ij} \right) a_j = \sum_{j=1}^{200} c_j a_j \ .$$

Each coefficient $c_j = \sum_{i=0}^{99} m_i e_{ij}$ is an integer between 0 and $\sum_{i=0}^{99} e_{ij} \leq 100 = \log n$, and thus $(c_1,\ldots,c_{200})$ is a legal solution of the knapsack system. For technical reasons, it is convenient to choose the $[e_{ij}]$ matrix in such a way that the number of 1's in each column ($\sum_{i=0}^{99} e_{ij}$) is exactly 63, since then each c_j can be represented as a 6-bit integer (whose average value is 31.5). This size restriction can be made an extra condition that valid signatures must satisfy, besides satisfying equation (1).

An intuitive way of looking at the signing procedure is to consider each row of the $[e_{ij}]$ matrix as a 200-tuple of 0's and 1's, and to add together (componentwise) all the rows corresponding to 1 bits in the binary representation of m. Note that the values of the a_j are not used in this process —all we need is the $[e_{ij}]$ matrix stored in memory.

This signing procedure is insecure, since anyone who has enough examples of message-signature pairs can find the $[e_{ij}]$ matrix, and thus forge arbitrary messages. The reason is that each time A signs a message m, he reveals two hundred linear

equations

$$(4) \qquad c_j = \sum_{i=0}^{99} m_i e_{ij} \qquad 1 \le j \le 200$$

in which the m_i and the c_j are known (from the given pair), and the only unknowns are the 20,000 e_{ij}. When 20,000 such equations have been accumulated, they can be solved and the e_{ij} can be found.

2.5. How to sign properly.

To solve the insecurity problem, we randomize the bits of m before we sign it, so that both the m_i and the e_{ij} in equation (4) become unknown. Each message-signature pair thus introduces 200 fresh variables m_i into the (non-linear!) system of equations, and thus the number of equations always lags behind the number of unknowns.

There are many ways in which m's bits can be randomized, but perhaps the simplest is to subtract from m a randomly chosen subset of the a_j's :

$$(5) \qquad m' = m - \sum_{j=1}^{200} \delta_j a_j \qquad (\text{mod } n)$$

(where each δ_j is 0 or 1), and then use the method of section 2.4 to sign the bits of m' instead of the bits of m:

$$(6) \qquad m' = \sum_{j=1}^{200} c_j' a_j \qquad (\text{mod } n) .$$

The signature of m' can now be easily transformed back into a signature of m by combining (5) and (6):

$$(7) \qquad m = \sum_{j=1}^{200} (c_j' + \delta_j)a_j = \sum_{j=1}^{200} c_j a_j \pmod{n} .$$

The chance that some coefficient $c_j = c_j' + \delta_j$ will overflow its six bit size is very small; even if it does, we just try again.

The reason we add to m a big subset of the a_j's is that we want to randomize its bits in a completely unpredictable way before we sign it. If the number of possible m' to which m could be transformed was small, a cryptanalyst could successively try all of them when using the equational method of the previous section. The number we subtract from m is not exactly a uniformly distributed random number, since the probability of subtracting r is proportional to the number of different subsets of the a_j's which sum up to r. However, this probability distribution is so hard to analyze that it seems to be impossible to exploit its slight non-uniformity in order to infer the value of m' from that of m.

3. Security considerations.

3.1. Cryptanalytic approaches.

The following list of four cryptanalytic approaches is illustrative, but certainly not exhaustive:

1) Static analysis: Without having any concrete examples of A's signatures, the cryptanalyst C might try to analyze the published data (i.e., the numbers $a_1,\ldots,a_{200}$ and n) in order to discover its hidden structure. As noted in section 2.3, it seems extremely unlikely that C will be able to find the $[e_{ij}]$ matrix or any other quick way of signing messages.

2) Dynamic analysis: C might try to forge A's signature on a new message m by combining known signatures on other messages. For example, if m can be written as the numerical sum of two messages m_1 and m_2, which A had previously signed as $(c_1^1,\ldots,c_{200}^1)$ and $(c_1^2,\ldots,c_{200}^2)$, respectively, then $(c_1^1 + c_1^2,\ldots,c_{200}^1 + c_{200}^1)$ might be a legal signature of m. To be legal, each $c_j^1 + c_j^2$ must be a 6 bit integer. When a larger number of signatures (say, a few tens) are added or subtracted, it becomes very hard to keep all the coefficients in their 0-63 interval simultaneously. Therefore even if C has a complete set of legal signatures of the powers of 2, he would not be able to use them to sign messages with more than a few l bits. In addition, we shall usually compactify messages before signing them (i.e., both the signer and the verifier will use some standard length-reducing and hard-to-invert function in order to transform arbitrarily

long messages into single 100-bit numbers, so that their signatures remain short). In compact forms, even messages which differ in a single bit become completely different, and thus C cannot hope to "compose" a desired forged message from a small number of known messages.

3) Planted messages: If C can cause A to sign certain special messages (using an unfaithful employee of A or otherwise), he might hope to benefit from watching the resultant signatures. For example, in the simple signing procedure of section 2.4, the signature of a message of the form 2^i immediately reveals a complete row of $[e_{ij}]$. Due to the randomization process, it is very unlikely that any of these messages will ever be signed as m' (regardless of what the original message m was), and thus there seem to be no "dangerous messages" from which the cryptanalyst can benefit. As a further precaution, it might be useful to re-randomize instead of signing any m' in which the number of 1 bits is under 35 or over 65. This guarantees that every signature $(c_1,\ldots,c_{200})$ is the sum of many, but not most, of the rows of $[e_{ij}]$.

4) Statistical analysis: This approach seems to be the most viable way of discovering the $[e_{ij}]$ matrix. Since each signature $(c_1,\ldots,c_{200})$ is the sum of a randomly chosen subset of matrix rows, we can statistically analyze large collections of signatures in order to find the structure of the matrix. The analysis concentrates mainly on the

correlations between the values of the c_j's in order to discover successively larger patterns of 0's and 1's in $[e_{ij}]$. This analysis is quite subtle, and there seem to be many ways in which it can be led astray.

3.2. Variations which might increase the security of the system.

In this section we mention three methods by which the structure of the signatures can be made more obscure and harder to analyze.

1) The δ_j in the randomization process can be chosen according to some non-uniform distribution, in which the various δ_j are strongly correlated. When added to the signature $(c'_1,\ldots,c'_{200})$ of m', the δ_j introduce irrelevant correlations which are not generated by the $[e_{ij}]$ matrix. If the distribution of the δ_j values is kept secret, the statistical methods can become unreliable. In order to strengthen this effect, it might be necessary to allow bigger values of δ_j (say, between 0 and 15).

2) In any cryptanalytic system, it is advisable to change the keys from time to time, since it reduces the chance of successful cryptanalytic attacks. This technique is particularly useful in signature systems, since a key discovered by the cryptanalyst after it has been replaced is quite useless (unlike privacy-ensuring systems, in which the replaced keys must be kept secret until the messages themselves can be made public).

In military applications, keys are usually replaced after a few days' use. Such a frequent change is quite inconvenient in big commercial public key networks, since the signature directory has to be updated and consulted constantly. We now show that in the system proposed in this paper, it is possible to benefit from frequent "key changes" without changing any of the numbers published in the directory.

The main idea is that the signature procedure uses the secret $[e_{ij}]$ matrix, rather than the published numbers a_j. Since the system of equations (2) is highly degenerate, the same published numbers can be generated by many other matrices. The signing machine can decide at any time to switch to a new such matrix, without notifying the directory or even its operator; the generated signatures will be determined by the new matrix and will thus have a new and different statistical behavior, but the verification procedure will remain unchanged. Typically, we shall change the published numbers once or twice a year, and "unofficially" change the matrix once a day (or after a predetermined number of signatures are generated).

One way of obtaining such a two-level system is to use the original $[e_{ij}]$ matrix as a seed which enables us to grow new matrices at will. Let $[E_{ij}]$ be a 100×200 matrix whose i^{th} row is some signature of 2^i in the

$[e_{ij}]$ system (due to the randomization process, there is an almost inexhaustible supply of distinct $[E_{ij}]$ matrices). It is easy to verify that $(a_1,\ldots,a_{200})$ remains a solution of equation (2) when $[e_{ij}]$ is replaced by $[E_{ij}]$, and thus messages can be signed and verified in the new $[E_{ij}]$ system without changing the published numbers. The only significant difference is that the entries in this matrix are six-bit integers instead of 0's and 1's, and thus correspondingly larger (12-bit instead of 6-bit) coefficients must be allowed in the signatures.

The security of the system is based on the fact that we use relatively few $[E_{ij}]$ matrices, each one of which is used to sign relatively few messages (by "relatively few" we mean $o(\sqrt{n})$, where n is the total number of messages signed). This implies that the statistical cryptanalytic methods are less likely to succeed in finding any of the $[E_{ij}]$ matrices, or in finding the common seed $[e_{ij}]$ used in their generation (for this part we may even assume that all the $[E_{ij}]$ matrices are known).

3) Instead of using one $[E_{ij}]$ matrix at a time, we can use two of them simultaneously. For each 2^i we now have two possible representations, given by the i^{th} row of the first matrix and the i^{th} row of the second matrix.

When we sign a message, we choose for its i^{th} bit the first or second representation with probabilities P_i and $1 - P_i$, respectively. This system behaves statistically as if we were using a single matrix whose rows are the weighted mean of the rows of the two matrices. While either one of these two matrices can be used in order to forge signatures, their weighted mean cannot. Note that the entries of this matrix can be arbitrary real numbers rather than integers, which makes the task of discovering it much more complicated.

4. Conclusions.

There are two main problems in public-key communications: privacy and signature generation. The Merkle-Hellman system and our system are complementary in the sense that they solve the first and second problem, respectively. Both systems are based on the knapsack problem, but they use differently structured keys. Most operations in these two systems are modular additions, which require little hardware and can be performed fast.

The main open problem in the proposed signature system concerns its security. Some specific questions are:

(i) Which cryptanalytic approaches have a chance to succeed?

(ii) How complicated are they and what are the resources they require?

(iii) What is the relation between the size of the knapsack

system and its security?

(iv) How should the user choose his knapsack system? How can he test the security of his particular choice?

(v) For how long can a chosen knapsack system be used?

(vi) Can arbitrary signatures be forged without knowing the $[e_{ij}]$ matrix?

(vii) Which precautions should the signer take when signing messages?

Acknowledgements: Numerous discussions with Ron Rivest and Len Adleman contributed to the development of this scheme.

Bibliography

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. Information Theory IT-22, 6, November 1976.

[2] M. Rabin, "The Signature Machine (SIM)", unpublished memo.

[3] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Signatures and Public-Key Cryptosystems", CACM 21, 2, February 1978.

[4] R. Merkle and M. Hellman, "Hiding Information and Receipts in Trap Door Knapsacks", to appear in IEEE Trans. Information Theory.

[5] R. Karp, "Reducibility Among Combinatorial Problems", in "Complexity of Computer Computations" (ed. R. Miller and J. Thatcher), Plenum Press, New York.