Op. 52

Constructing Digital Signatures from a One Way Function

Leslie Lamport

Computer Science Laboratory

SRI International

18 October 1979

CSL - 98

333 Ravenswood Ave. • Menlo Park, California 94025
(415) 326-6200 • Cable: SRI INTL MPK • TWX: 910-373-1246

1. Introduction

A digital signature created by a sender P for a document m is a data item $\sigma_P(m)$ having the property that upon receiving m and $\sigma_P(m)$, one can determine (and if necessary prove in a court of law) that P generated the document m .

A one way function is a function that is easy to compute, but whose inverse is difficult to compute [1]. More precisely a one way function ϕ is a function from a set of data objects to a set of values having the following two properties:

1. Given any value v , it is computationally infeasible to find a data object d such that $\phi(d) = v$.
2. Given any data object d , it is computationally infeasible to find a different data object d' such that $\phi(d') = \phi(d)$.

If the set of data objects is larger than the set of values, then such a function is sometimes called a one way hashing function.

We will describe a method for constructing digital signatures from such a one way function ϕ . Our method is an improvement of a method devised by Rabin [2]. Like Rabin's, it requires the sender P to deposit a piece of data α in some trusted public repository for each document he wishes to sign. This repository must have the following properties:

- α can be read by anyone who wants to verify P's signature.
- It can be proven in a court of law that P was the creater of α .

Once α has been placed in the repository, P can use it to generate a signature for any single document he wishes to send.

Rabin's method has the following drawbacks not present in ours.

1. The document m must be sent to a single recipient Q , who then requests additional information from P to validate the signature. P cannot divulge any additional validating information without compromising information that must remain private to prevent someone else from generating a new document m' with a valid signature $\sigma_P(m')$.
2. For a court of law to determine if the signature is valid, it is necessary for P to give the court additional private information.

This has the following implications.

- P -- or a trusted representative of P -- must be available to the court.
- P must maintain private information whose accidental disclosure would enable someone else to forge his signature on a document.

With our method, P generates a signature that is verifiable by anyone, with no further action on P's part. After generating the signature, P can destroy the private information that would enable someone else to forge his signature. The advantages of our method over Rabin's are illustrated by the following considerations when the signed document m is a check from P payable to Q .

1. It is easy for Q to endorse the check payable to a third party R by sending him the signed message "make m payable to R ". However, with Rabin's scheme, R cannot determine if the check m was really signed by P , so he must worry about forgery by Q as well as whether or not P can cover the check. With our method, there is no way for Q to forge the check, so the endorsed check is as good as a check payable directly to R signed by P . (However, some additional mechanism must be introduced to prevent Q from cashing the original check after he has signed it over to R .)
2. If P dies without leaving the executors of his estate the information he used to generate his signatures, then Rabin's method cannot prevent Q from undetectably altering the check m -- for example, by changing the amount of money payable. Such posthumous forgery is impossible with our method.
3. With Rabin's method, to be able to successfully challenge any attempt by Q to modify the check before cashing it, P must maintain the private information he used to generate his signature. If anyone (not just Q) stole that information, that person could forge a check from P payable to him. Our method allows P to destroy this private information after signing the check.

2. The Algorithm

We assume a set $\underline{M}$ of possible <u>documents</u>, a set $\underline{K}$ of possible <u>keys</u>[1],

[1]The elements of $\underline{K}$ are not keys in the usual cryptographic sense, but are arbitrary data items. We call them keys because they play the same role as the keys in Rabin's algorithm.

and a set $\underline{V}$ of possible values. Let Σ denote the set of all subsets of $\{1, \ldots, 40\}$ containing exactly 20 elements. (The numbers 40 and 20 are arbitrary, and could be replaced by $2n$ and n. We are using these numbers because they were used by Rabin, and we wish to make it easy for the reader to compare our method with his.)

We assume the following two functions.

1. A function $F : \underline{K} \rightarrow \underline{V}$ with the following two properties:

   a. Given any value v in $\underline{V}$, it is computationally infeasible to find a key k in $\underline{K}$ such that $F(k) = v$.

   b. For any small set of values $v_1, \ldots, v_m$, it is easy to find a key k such that $F(k)$ is not equal to any of the v_i.

2. A function $G : \underline{M} \rightarrow \Sigma$ with the property that given any document m in $\underline{M}$, it is computationally infeasible to find a document $m' \neq m$ such that $G(m') = G(m)$.

For the function F, we can use any one way function ϕ whose domain is the set of keys. The second property of F follows easily from the second property of the one way function ϕ. We will discuss later how the function G can be constructed from an ordinary one way function.

For convenience, we assume that P wants to generate only a single signed document. Later, we indicate how he can sign many different documents. The sender P first chooses 40 keys k_i such that all the values $F(k_i)$ are distinct. (Our second assumption about F makes this easy to do.) He puts in a public repository the data item $\alpha = (F(k_1), \ldots, F(k_{40}))$. Note that P does not divulge the keys k_i, which by our first assumption about F cannot be computed from α.

To generate a signature for a document m, P first computes $G(m)$ to obtain a set $\{i_1, \ldots, i_{20}\}$ of integers. The signature consists of the 20 keys $k_{i_1}, \ldots, k_{i_{20}}$. More precisely, we have $\sigma_P(m) = (k_{i_1}, \ldots, k_{i_{20}})$, where the i_j are defined by the following two requirements:

(i) $G(m) = \{i_1, \ldots, i_{20}\}$.

(ii) $i_1 < \ldots < i_{20}$

After generating the signature, P can destroy all record of the 20 keys k_s with s not in G(m) .

To verify that a 20-tuple $(h_1, \ldots, h_{20})$ is a valid signature $\sigma_P(m)$ for the document m , one first computes G(m) to find the i_j and then uses α to check that for all j , h_j is the i_jth key. More precisely, the signature is valid if and only if for each j with $1 \leq j \leq 20$: $F(h_j) = \alpha_{i_j}$, where α_i denotes the ith component of α , and the i_j are defined by the above two requirements.

To demonstrate that this method correctly implements digital signatures, we prove that it has the following properties.

1. If P does not reveal any of the keys k_i , then no-one else can generate a valid signature $\sigma_P(m)$ for any document m .
2. If P does not reveal any of the keys k_i except the ones that are contained in the signature $\sigma_P(m)$, then no-one else can generate a valid signature $\sigma_P(m')$ for any document $m' \neq m$.

The first property is obvious, since the signature $\sigma_P(m)$ must contain 20 keys k_i such that $F(k_i) = \alpha_i$, and our first assumption about F states that it is computationally infeasible to find the keys k_i just knowing the values $F(k_i)$.

To prove the second property, note that since no-one else can obtain any of the keys k_i , we must have $\sigma_P(m') = \sigma_P(m)$. Moreover, since the α_i are all distinct, for the validation test to be passed by $\sigma_P(m')$, we must also have $G(m') = G(m)$. However, our assumption about G states that it is computationally infeasible to find such a document m' . This proves the correctness of our method.

For P to send many different documents, he must use a different α for each one. This means that their must be a sequence of 40-tuples $\alpha_1, \alpha_2, \ldots$ and the document must indicate which α_i is used to generate that document's signature. The details are simple, and will be omitted.

3. Constructing the Function G

One way functions have been proposed whose domain is the set of documents and whose range is a set of integers of the form $\{0, \ldots, 2^n-1\}$, for any reasonably large value of n. (It is necessary for n to be large enough to make exhaustive searching over the range of ϕ computationally infeasible.) Such functions are described in [1] and [2]. The obvious way to construct the required function G is to let ϕ be such a one way function, and define $G(m)$ to equal $R(\phi(m))$, where $R : \{0, \ldots, 2^n-1\} \rightarrow \Sigma$.

It is easy to construct a function R having the required range and domain. For example, one can compute $R(s)$ inductively as follows:

1. Divide s by 40 to obtain a quotient q and a remainder r
2. Use r to choose an element x from $\{1, \ldots, 40\}$. (This is easy to do, since $0 \leq r \leq 40$.)
3. Use q to choose 19 elements from the set $\{1, \ldots, 40\} - \{x\}$ as follows:

    a. Divide q by 39 to obtain a quotient ...

It requires a careful analysis of the properties of the one way function ϕ to be sure that the resulting function G has the required property. We suspect that for most one way functions ϕ, this method would work. However, we cannot prove this.

The reason constructing G in this manner might not work is that the function R from $\{0, \ldots, 2^n\}$ into Σ is a many to one mapping, and the resulting "collapsing" of the domain might defeat the one way nature of ϕ. However, it is easy to show that if the function R is one to one, then property (ii) of ϕ implies that G has the required property. To construct G, we need only find an easily computable one to one function R from $\{0, \ldots, 2^n-1\}$ into Σ, for a reasonably large value of n.

We can simplify our task by observing that the function G need not be defined on the entire set of documents. It suffices that for any document

m, it is easy to modify m in a harmless way to get a new document that is in the domain of G. For example, one might include a meaningless number as part of the document, and choose different values of that number until he obtains a document that is in the domain of G. This is an acceptable procedure if (i) it is easy to determine whether a document is in the domain, and (ii) the expected number of choices one must make before finding a document in the domain is small.

With this in mind, we let $n = 40$ and define $R(s)$ as follows: if the binary representation of s contains exactly 20 ones, then $R(s) = \{i :$ the ith bit of s equals one$\}$, otherwise $R(s)$ is undefined. Approximately 13% of all 40 bit numbers contain exactly 20 ones. Hence, if the one way function ϕ is sufficiently randomizing, there is a .13 probability that any given document will be in the domain of G. This means that randomly choosing documents (or modifications to a document), the expected number of choices before finding one in the domain of G is approximately 8. Moreover, after 17p choices, the probability of not having found a document in the domain of G is about $1/10^p$. (If we use 60 keys instead of 40, the expected number of choices to find a document in the domain becomes about 10, and 22p choices are needed to reduce the probability of not finding one to $1/10^p$.)

If the one way function ϕ is easy to compute, then these numbers indicate that the expected amount of effort to compute G is reasonable. However, it does seem undesirable to have to try so many documents before finding one in the domain of G. We hope that someone can find a more elegant method for constructing the function G, perhaps by finding a one to one function R which is defined on a larger subset of $\{0, \ldots, 2^n\}$.

Note: We have thus far insisted that $G(m)$ be a subset of $\{1, \ldots, 40\}$ consisting of exactly 20 elements. It is clear that the generation and verification procedure can be applied if $G(m)$ is any proper subset. An examination of our correctness proof shows that if we allow $G(m)$

to have any number of elements less than 40, then our method would still have the same correctness properties if G satisfies the following property:

- For any document m , it is computationally infeasible to find a different document m' such that G(m') is a subset of G(m) .

By taking the range of G to be the collection of 20 element subsets, we insure that G(m') cannot be a proper subset of G(m) . However, it may be possible to construct a function G satisfying this requirement without constraining the range of G in this way.

REFERENCES

[1] Diffie, W. and Hellman, M. "New Directions in Cryptography". _IEEE Trans. on Information Theory IT-22_ (November 1976), 644-654.

[2] Rabin, M. "Digitalized Signatures", in _Foundations of Secure Computing_, Academic Press (1978), 155-168.