INTERACTIVE IDENTIFICATION AND DIGITAL SIGNATURES

Ernest F. Brickell and Kevin S. McCurley

Ernest F. Brickell *is a supervisor, and* ***Kevin S. McCurley*** *is a senior member of technical staff. They are in the Theoretical Computer Science Division of the Department of Mathematics and Computational Science at Sandia National Laboratories in Albuquerque, New Mexico. Mr. Brickell's group works on cryptographic algorithms and protocols, cryptanalysis, discrete algorithms, and discrete mathematics. He joined AT&T/Sandia in 1981, went to Bell Communications Research in 1985, and returned to Sandia in 1988. Mr. Brickell has a B.S. in mathematics from Oklahoma State University (Stillwater), and both an M.S. in computer science and a Ph.D. in mathematics from Ohio State University (Columbus). Mr. McCurley's primary research interests are in cryptology, discrete algorithms, and parallel computation. He joined Sandia in 1989* (continued on page 86)

An interactive identification scheme is a method for remotely (or locally) verifying an individual's identity. Interaction prevents the person who is doing the verifying from collecting information that could be used later to impersonate the individual. It also prevents anyone else from gaining such information. Interactive identification schemes have many potential applications—from providing more secure remote computer logins to providing more secure bank-card transactions. Such schemes can also be used to construct digital signatures, a way of signing an electronic document that makes the signature easy to check but difficult for an unauthorized person to forge. In this paper, we describe the general problems of identification and signatures, explain some shortcomings of current methods, and discuss how interactive identification schemes address them. We also describe an interactive identification scheme discovered at Sandia National Laboratories in which the security is based on the computational difficulty of solving two hard problems.

Introduction

Because our society has come to depend more and more on the electronic storage and transmission of information, we now face a pressing need for new techniques to protect the integrity of this information. The three basic needs are:

- *Privacy.* Proprietary information and personal records provide examples of information whose sensitivity requires protection against unauthorized dissemination. Protection methods that rely on restrictions to physical access can be costly; but more important, they also severely restrict the way in which the information can be used legitimately. An alternative is to encrypt the information.
- *Identification.* When information is to be passed between parties

Panel 1. Abbreviations, Acronyms, and Terms

α — an integer the KAC chooses; $\alpha^q \equiv 1 \pmod{p}$

$b \equiv a \pmod{n}$ — notation used to show that n divides $b - a$

ACM — Association for Computing Machinery

DES — Digital Encryption Standard

digital signature — a piece of data the signer creates to provide evidence he approved an electronic document. This signature may be part of the message or a totally separate piece of information.

DNA — deoxyribonucleic acid

DSS — Digital Signature Standard; recently proposed by the National Institute of Standards and Technology

e — a random number

$h(x, m)$ — the signature scheme's hash function, where x is a computed value and m is the message

I — the prover's private identification string, which might consist of his name and other identification information that might be required or useful

K — a value the prover computes and uses as his public key; for the Sandia scheme, $K \equiv \alpha^{-S} \pmod{p}$

k — security parameter selected by the KAC

KAC — key authentication center

m — the message to be signed (in a signature scheme)

MD4 — message digest algorithm; a hash function developed by Ronald L. Rivest of MIT

MIT — Massachusetts Institute of Technology

NIST — National Institute of Standards and Technology

p — a prime number

paperless society — a society in which all information is stored or retrieved only in electronic form

prover — the person being identified

q — a prime number

r — a random number the prover chooses; $r \in \{1, \cdots, p - 1\}$

RSA — encryption and signature scheme invented at MIT by Ronald L. Rivest, Adi Shamir, and Leonard Adleman

S — a random number the prover chooses as his private key; for the Sandia scheme, $S \in \{1, \cdots, p - 1\}$

$\mathcal{S}$ — the KAC's signature of the prover's public key K

Snefru — a one-way hash function developed by Ralph Merkle of Xerox PARC

t — a security parameter

type-I error — the verifier may fail to accept the proof of identity from a legitimate prover.

type-II error — the verifier may accept an illegitimate proof of identity from an illegitimate prover.

verifier — the person who is verifying another person's identity

witness — the secret key or value known by the prover

witness hiding — the identification scheme does not reveal any information about the witness.

x — a computed value

y — an integer the prover computes, such that $y \equiv r + Se \pmod{p - 1}$

(i.e., people or machines) who are physically isolated from each other or have never before met, they often need to verify each other's identity.

- *Signatures.* Traditionally, a paper document's authenticity has been offered some form of protection through written signatures, watermarks, or seals. Unfortunately, these techniques offer no protection for information that exists only in electronic form. For such information, we need the concept of a *digital signature.*

In this paper, we discuss solutions to the last two problems, namely, identification and digital signatures. Our goal is to describe some recently invented mathematical techniques and the motivation for their development. (Panel 1 defines acronyms and terms used in this paper.)

The Need for Identification. Loosely speaking, identification is a process by which a party (called the *verifier*) becomes convinced that she recognizes another party (called the *prover*). To reduce ambiguity in this paper's discussions, we will use female pronouns to refer to the verifier and male pronouns to refer to the prover.

Later, we will discuss, in general terms, some of the approaches that can be used for identification. But before we examine the different methods, let us consider some cases where identification is used:

- An employee comes to work and tries to gain entrance to his place of employment.
- A person attempts to login to a computer that is located many miles away.
- A computer attempts to request a service that is provided by another machine on a network.
- A person involved in armed conflict attempts to convince other forces that he is on their side of the conflict. (This situation is commonly referred to as *identify friend or foe*).
- A person uses a charge card. The business asks him for identification to prove that he is authorized to use the card, and also inspects the card for authenticity.
- During on-site verification under a nuclear-weapons treaty, the verifier may need to be convinced that the weapons she has just seen are indeed ones that were registered before. Here, the prover is the weapon.

In each case, the job of the prover is to offer evidence of his identity to the verifier, and the role of the verifier is to examine the evidence and form an opinion about its accuracy. Several methods can be used for this identification, and will be discussed shortly.

The Need for Digital Signatures

The problem of authenticating documents bears some similarity to the problem of identification. Here, the role of a handwritten signature on a paper document is to provide evidence that this person approved the document. If we are to realize the full potential of a *paperless society*, then we need methods to ensure the validity of information that exists only in an electronic form.

In a seminal paper of 1976,[1] Diffie and Hellman were the first to discuss the notion of a *digital signature*, i.e., a piece of data the signer creates to provide evidence that he approved the electronic document. A digital signature may be either part of the message or a totally separate piece of information.

Ideally, the signature for such an electronic message should have the following properties:

- The signature should be easy for the legitimate user to create.
- The signature should be difficult for an illegitimate user to create.
- The signature's validity should be easy for anyone to verify.
- The signature should be dependent on the message that it signs (i.e., connected in some way to that message and to no other), so it can reveal any tampering with the original document.
- It should be difficult for anyone to create a different document for which the signature is valid.

While there is a natural connection between the notions of identification and authentication, it is not immediately clear how they are related. Later, we will describe how some identification methods that were discovered recently can be used to provide digital signatures. But first, we will examine some basic properties of identification schemes, which will lead naturally to the consideration of *interactive* identification schemes. Then, we will return to the problem of constructing digital signatures.

Traditional Approaches to Identification

An identification protocol may fail in two ways. To describe such failures, we use standard terminology from the theory of statistical hypothesis testing:

- *Type-I error.* The verifier may fail to accept the proof of identity from a legitimate prover.
- *Type-II error.* The verifier may accept an illegitimate proof of identity from an illegitimate prover.

The threats posed by each of these breaches of the protocol may have different consequences, depending on the situation. For example, in business transac-

tions, the verifier may be willing to accept a few type-II errors in exchange for less inconvenience imposed on customers. On the other hand, when controlling access to a highly sensitive facility, the risk of a type-II error may outweigh any inconvenience imposed on a prover who is seeking access.

Because of these and other factors, a wide variety of techniques have been developed for proving identity. All the identification techniques that we know of use one or more of four methods: passwords, physical characteristics, physical objects, or actions.

Passwords. A password is a piece of information that is known only by the prover and verifier. Ideally, a password should consist of random data, so that an impersonator will have no better method than to guess from a large set of possibilities.

Unfortunately, people are not good at remembering random data. Often, they write down the password and keep that piece of paper in an accessible location. Or, they circumvent the randomness of the password by choosing data that is easy for them to remember and for an opponent to guess.

Another problem with passwords occurs when the prover and verifier are communicating via a communications channel that is subject to eavesdropping. If the password is transmitted over the line without encryption, then the eavesdropper will be able to impersonate the prover at a later time. This, for example, can be a serious problem with computer logins over a network.

One strategy that avoids the transmission of passwords "in the clear" (i.e., not encrypted) is used in the Kerberos authentication system.[2] This system is based, in turn, on the trusted-third-party model of authentication proposed by Needham and Schroeder.[3] Unfortunately, the introduction of a trusted third party requires real-time communication with the trusted third party and introduces other problems as well,[4] which in turn limits the applicability of the model.

One last problem with passwords is that the verifier also knows the prover's password. (We are using the word *know* in a broad sense here. For instance, a UNIX® operating system knows the user's password during the login procedure, even though only an encrypted form of the password is stored. UNIX is a registered trademark of UNIX System Laboratories, Inc.) If the prover uses the same password to prove his identity to another verifier, then there is nothing to prevent the first verifier from impersonating the prover. The problem is compounded because people who have trouble remembering passwords often use a single password for several different situations.

Physical Characteristics. A prover may have a unique physical characteristic that can be used for identification purposes. For example, humans possess such biometric information as fingerprints, weight, blood-vessel patterns on the retina, DNA patterns, and hand geometry. (Weight is often used in combination with other factors. For example, you may be required to stand on a scale while your hand geometry is verified.)

Although physical characteristics are usually difficult to duplicate, they are not foolproof. Any machine that measures them will generally have a limit to its tolerances (i.e., the range of variation that is considered acceptable in measurements).

For several years, James Holmes, Russell Maxwell, and Larry Wright of Sandia National Laboratories have carried out an extensive evaluation program of available systems that rely on these techniques. Their findings show[5] that the technologies to measure these factors have advanced rapidly in recent years, and are now useful for some identification purposes.

A group at Sandia recently developed a method to create a unique physical characteristic on an object. What motivated this work was the requirement for verification in arms-control treaties, where either side needs to verify that a specific weapon was authorized. In the method developed at Sandia, reflective particles embedded in a transparent adhesive are applied to the surface of the object to be identified. The pattern of reflection from the particles provides a characteristic that is easy

to read and virtually impossible to duplicate.

Objects. A physical object (e.g., a token or physical key) in a person's possession can act as an agent for the prover. A token may be simply a storage device such as a magnetic card, or may perform some action for the prover. Active tokens can be extremely useful in an identification system because they can perform calculations for the prover.

Actions. An identification system may require an action that only the prover can perform (e.g., a handwritten signature).

Interactive Identification Schemes

In an interactive identification scheme, the prover establishes his identity by communicating with the verifier until she is convinced that he knows a certain piece of information that only the real prover could know. There is no need to convey secret information, only information that convinces the verifier that the prover possesses the secret information.

This solves several of the major problems with identification schemes that we described in this paper's introduction. For example, suppose the prover and the verifier share a secret encryption key. When the verifier issues a random challenge to the prover, he encrypts that message and sends the encrypted version back to her. She can then check that the response matches her encrypted value for the random challenge, which provides evidence that the prover possesses the secret key. A system[6] based on this approach is now in place at AT&T Bell Laboratories to control access to laboratory computers that are connected to the Internet. [*Internet*, the largest interconnected network in the world, consists of large national backbone networks (commercial, businesses, and military) and regional and local campus networks.] Several vendors also sell commercial systems that are based on this principle.

Although this approach can function well in an environment where the prover repeatedly proves his identity to the same verifier, it still requires a secret key for each prover-verifier pair. Another problem arises because the keys that the verifier stores must be protected to prevent others from reading them. We will describe a new interactive identification scheme developed at Sandia that does not require the verifier to store any secret information. We have discovered mathematical proof that, with this scheme, no eavesdropper can learn anything that will help him impersonate the prover, at least under some reasonable assumptions about computational intractability.

For an interactive identification scheme to be useful, it must protect against type-I and type-II errors. The properties that an interactive identification scheme should have are commonly categorized as follows:

- *Completeness.* If the prover and verifier are both honest, then with very high probability the verifier will accept the proof of identity.
- *Soundness.* If a prover does not know the secret key associated with the identification submitted and the verifier is honest, then with very high probability the verifier will reject the proof of identity.
- *Witness hiding.* If the prover is honest, then nobody should learn information that will allow him to impersonate the prover, whether the verifier is honest or dishonest, even if the protocol is repeated many times. The secret known by the prover is often referred to as a *witness*. The term *witness hiding*, introduced by Feige and Shamir,[7] describes a scheme that does not reveal any information about the witness.

A scheme clearly protects against type-I errors if it satisfies the completeness criterion, and it will protect against type-II errors if it also satisfies the criteria of soundness and witness hiding.

The secret information known to the prover will be certified by a centralized authority, which we will call a *key authentication center* (hereafter abbreviated as the KAC). To enroll in the system, the prover would go to the KAC and produce accepted, legal evidence of his identity. Then, the prover (perhaps, in collaboration with the KAC) would produce a public identification key, K, and his

private identification key, S. Finally, the KAC would form an identification string, I, for the prover, which might consist of his name and other identification information that might be required or useful.

When a prover presents his identification string, I, and his public key, K, to a verifier, she must be convinced that K really is the public key for the person identified by I. Several methods have been proposed for doing this. In the simplest method, I and K satisfy an equation, i.e., $f(I, K) = 0$ for some function f. This is called an *identity-based scheme*. The Fiat–Shamir scheme[8] can be implemented as an identity-based scheme.

For other identification schemes, it has not been possible to define a functional relation between I and K. Therefore, other alternatives must be used to certify I and K. One option is for the KAC to enter the pair (I, K) into a certifiable and publicly available directory. Another option is for the KAC to use a digital signature scheme to produce a signature $S(I, K)$, which is then given to the prover. In the remainder of this paper, we will assume that the latter method is being used, but all protocols could be easily modified for the public-directory method of certification.

Notice that we are using a trusted third party, i.e., the KAC, to authenticate keys. However, its usage here is considerably different from that used in the Kerberos approach, because we use the third party only once—i.e., when the user is initially registered.

Later in this section, we will describe the interactive identification scheme we discovered and will compare its performance and level of provable security to some earlier schemes. This comparison will show that none of the schemes is superior under all conditions, and that they are all practical for some applications.

In comparing the security of these schemes, we consider whether it is possible to prove that the scheme is secure if we assume that a computational problem is infeasible to solve. (By *infeasible* we mean that, in theory, the problem can be solved but not in a reasonable amount of time.)

We will refer to the following specific computational problems in this paper:

- *Factoring.* Given an integer n, find the prime divisors of n.
- *Discrete logarithms.* Given integers n, g, and a, find an integer x such that $g^x \equiv a \pmod{n}$, provided one exists. [The notation $b \equiv a \pmod{n}$ is used to show that n divides $b - a$.]

These problems are widely believed to be computationally difficult, particularly when compared to the difficulty of verifying the answer.

For example, if we are given the prime numbers p and q that are the factors of an integer n, then we may easily recover n by simply multiplying p and q together. The inverse problem of computing p and q from n is what is believed to be hard. In practice, this problem is almost always infeasible when p and q are 100 decimal digits in size. For further information on the current state of the art in factoring, see Lenstra and Manasse[9] or Pomerance.[10]

The same is true for the problem of discrete logarithms, because $a = g^x \pmod{n}$ is relatively easy to compute when we are given g, x, and n, even if x is very large. (See Panel 2.) On the other hand, computing x from a using the best known algorithms is almost always infeasible if n has 200 decimal digits. For more information on computing discrete logarithms, see LaMacchia and Odlyzko[11] and McCurley.[12]

The Sandia Scheme. In the interactive identification scheme that we have proposed, a trusted KAC chooses a security parameter k (≈ 140), and primes p, q, and w such that:

- qw divides $p - 1$,
- q^2 does not divide $p - 1$,
- $q, w \geq 2^k$, and
- $p \geq 2^{512}$.

The KAC also chooses an integer α with $\alpha^q \equiv 1 \pmod{p}$. The KAC publishes p and α, but not q or w. At this point, the KAC can destroy the values q and w, because they will no longer be needed.

When a user wishes to join the system, he chooses a random number $S \in \{1, \cdots, p-1\}$ as his private key. He then computes $K \equiv \alpha^{-S} \pmod{p}$, and

Panel 2. Faster Exponentiation

One argument made against cryptographic schemes that require modular exponentiation is that these arithmetic operations are too slow for many implementations. Therefore, faster methods for exponentiation are of great interest. Recently, the authors and Daniel Gordon[26] discovered such a method for the case where a user must compute $g^r \bmod n$ for the same value of g and n, but for many different values of r. This development is recent, but may be protected in the future by a patent filed by Sandia National Laboratories.

The basic idea used by this method is to store a small number of precomputed powers of g from which $g^r \bmod n$ can be computed easily for an arbitrary value of r. As an example of this approach, suppose we need to compute $g^r \bmod n$ for a random r from the set $\{1, \cdots, 2^{160}\}$. (This might be the case for the proposed NIST signature standard, described in Panel 3.)

If we represent the exponent r in base 16 (i.e., in hexadecimal notation), then we can think of this as a representation:

$$r = \sum_{i=0}^{m} d_i 16^i,$$

where $0 \le d_i < 16$. Here, we can compute $g^r \bmod n$ from:

$$g^r = \prod_{i=0}^{m} g^{d_i 16^i} \bmod p$$

$$= \prod_{d=1}^{15} c_d^d \bmod p, \qquad (1)$$

where

$$c_d = \prod_{\substack{i=0 \\ d_i = d}}^{m} g^{b^i} \bmod p.$$

If we have precomputed the values $g^{16^i} \bmod p$, then we can compute the numbers c_d and, from these, compute the product (1) using only 28 multiplications modulo p. If we ignore the precomputation, this method allows us to compute $g^r \bmod p$ in 52 modular multiplications, on average. (The ordinary binary method requires 237 modular multiplications, on average.) Because the modular multiplications represent the majority of work done in the exponentiation, this method can yield significant improvements in speed.

A somewhat different approach can also be used for efficiently computing expressions of the form $g^r K^e \bmod p$, which is required in some of the protocols we discussed in this paper.

gives it to the KAC as his public key.

The interactive identification process involves the following steps:

1. The prover chooses a random number $r \in \{1, \cdots, p-1\}$, and computes $x \equiv \alpha^r \pmod p$. The prover then sends his identification string I, his public key K, the KAC's signature $S(I, K)$, and x to the verifier.
2. The verifier checks the authenticity of the prover's public key by verifying the signature $S(I, K)$. If the keys are authentic, she then chooses a random number $e \in \{1, \cdots, 2^t\}$ and transmits e to the prover.
3. The prover then computes an integer y such that $y \equiv r + Se \pmod{p-1}$, and sends y to the verifier.
4. The verifier checks that $x \equiv \alpha^y K^e \pmod p$ and, if this condition is satisfied, accepts the prover's proof of identity.

Figure 1 gives a graphic illustration of this protocol.

The major computation performed by the prover is the exponentiation required to determine x. However, he can do this computation off-line, well before the value is needed for an identification. The only on-line computation required of the prover is to determine y, which is a minor computation.

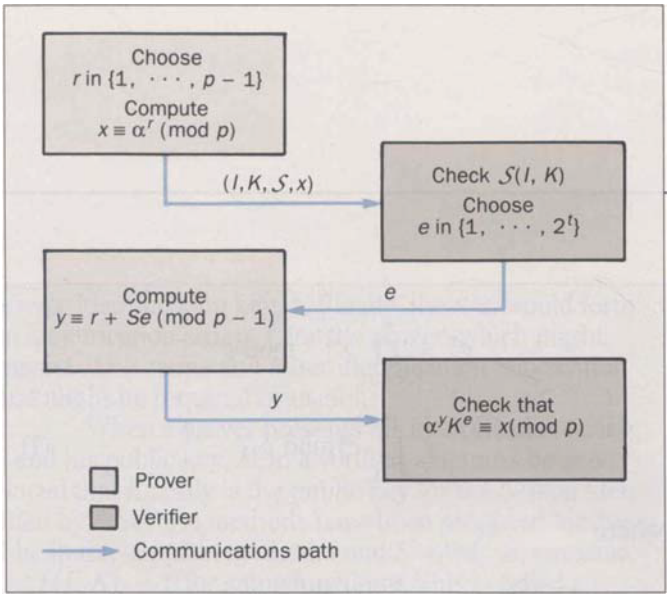

Figure 1. Sandia's interactive identification scheme. To prove his identity, the prover selects a random number *r* and computes *x*. He sends the identification string *I*, his public key *K*, the key authentication center's signature *S*, and *x* to the verifier. She checks that his public key is authentic and chooses a random number *e*, which she transmits to the prover. He uses it to compute the integer *y*, which he returns to the verifier. She uses it to check if *x* satisfies the condition and, if it does, accepts the prover's proof of identity.

The level of security required dictates the choice of t. For identification purposes, the probability that an illegitimate prover will be able to provide the proper response can be shown to be at most 2^{-t}. We, therefore, suggest a value of $t = 20$, which gives a chance of less than one in a million that a type-II error will occur.

One can easily see that our scheme is *complete*; that is, if the prover and verifier are both honest, then the prover will always be able to provide a y that the verifier will accept.

That our scheme is *sound* is not as trivial to show. We proved[13] the soundness by showing that, if the verifier accepts the proof of identity, then she is convinced that the prover has a high probability of knowing a discrete logarithm of K. Moreover, we proved that, if an impostor had a high probability of deceiving the verifier, then this impostor could join forces with the legitimate prover to factor $p - 1$. Hence, an illegitimate prover must be doubly powerful. To compute the secret information of a legitimate prover, he must be able to solve *two* hard problems; namely, compute a discrete logarithm and factor $p - 1$.

The proof that our scheme is *witness hiding* (i.e., it does not disclose the secret) is even more subtle. We have shown[13] that, if the scheme leaked enough information for an impostor to compute a witness (i.e., the discrete logarithm), then (as in the proof of soundness) the impostor and the legitimate prover could pool their knowledge to compute a nontrivial factor of $p - 1$. Hence, if factoring $p - 1$ is difficult, the protocol does not leak enough information to determine a witness.

The protocol that we have described is a modification of an earlier protocol by Schnorr.[14] In his protocol, Schnorr proposed that the factor q be revealed. Then, the prover would be able to pick r in $\{1, \cdots, q\}$ instead of in $\{1, \cdots, p - 1\}$, as in our scheme. As a result, Schnorr's scheme does not require as much computation as ours. However, he was not able to present any rigorous mathematical arguments to show that his scheme was witness hiding, and was only able to prove that the scheme was sound if computing discrete logarithms was hard.

Comparison of Identification Schemes. In this section, we compare the level of provable security, amount of computation, amount of data communicated, and storage used by the Sandia scheme with the resources used by several other schemes. For the comparison, we use the Schnorr scheme mentioned previously,[14] the Fiat–Shamir scheme,[15] the identity-based version of the Fiat–Shamir scheme,[8] and the Micali–Shamir scheme.[16] (We refer interested readers to the original sources for descriptions of the different schemes.)

Much of the work done on interactive identification schemes has been devoted to devising schemes that are both provably secure under some reasonable hypotheses and efficient to implement. So far, there appears to be a tradeoff between these two goals. The security for all the schemes we have discussed depends on the difficulty of solving a computational problem. While these

Table I. Efficiency of Identification Schemes

	Schnorr		Sandia		Fiat–Shamir		Micali–Shamir
Identity based?	no		no		yes	no	no
Provable witness hiding?	no		yes		no	yes	yes
Length of public key (bytes)	64		64		NA	1280	69
Length of secret key (bytes)	18		64		1280	1280	1280
Off-line modular multiplications by prover (number of operations)	207	47*	765	129*	1	1	1
Prover's auxiliary storage (bytes)	146	2450	128	6720	64	64	0
On-line modular multiplications by prover (number of operations)	< 1		< 1		10	10	10
Communications† (bytes)	89		135		135	135	135
Modular multiplications by the verifier (number of operations)	235	77*	793	157*	11‡	11	3
Verifier's auxiliary storage (bytes)	128	2450	128	6720	NA	NA	NA

NA = not applicable.
• This entry for number of modular multiplications for the Schnorr and Sandia schemes represents the result of using precomputed values (see Panel 2).
† The values given assume that a public certified directory is used to look up the public keys. If this is not true, then the total communication must be increased by the size of the public key plus the length of S, the KAC's signature of the prover's public key K. A reasonable assumption for the signature's size is 34 to 80 bytes.
‡ This does not count the time required to compute the public keys from the prover's identity.

assumptions appear reasonable today, there is no guarantee that a clever mathematician will not discover an efficient algorithm to factor integers or compute discrete logarithms.

This was one of our major motivations for developing the Sandia scheme, whose security is based on the difficulty of solving *two* problems that are currently intractable. Our goal was to provide an added measure of security against unpredictable algorithmic developments, because it seems unlikely that both of the problems would suddenly become easy to solve. None of the other schemes we describe have this property.

As the reader will notice from our comparisons, this added measure of security extracts a toll for some measurements of performance. However, for many applications, the performance of our scheme will be adequate, or the added measure of security makes it worthwhile to accept the performance degradation.

Finally, we have included the Schnorr scheme in the comparisons, even though it is the only one for which there is no reasonable hypothesis that leads to a proof of the witness hiding property.

The quantitative comparisons are based on a particular choice of security parameters and on a particular implementation of each system. For instance, in our scheme, the prover must compute an exponentiation with a fixed 512-bit modulus and a random 512-bit exponent. By the straightforward method described earlier for interactive identification schemes, this would take 765 multiplications on average. But because the

exponentiations are always over the same base, α, it is possible to save substantially by precomputing and storing α^i for well-chosen values of i. (See Panel 2.)

The verifier could do similar precomputation, and we include this in the table. Another option for the verifier is to use addition chains[17] to reduce the number of multiplications; we do not show this in the table. Similar comments apply to the Schnorr scheme.

In Table I, we compare each scheme's efficiency for the following choices of parameters:

- Schnorr: p has 512 bits, q has 140 bits, e has 20 bits.
- Sandia: p has 512 bits, q has 140 bits, e has 20 bits.
- Fiat–Shamir: n has 512 bits, $k = 20, t = 1$.
- Micali–Shamir: n has 512 bits, $k = 20, t = 1$.

In the table, the amount of data communicated between the parties and the amount of storage are given in bytes. For multiplications with the Schnorr and Sandia schemes, we give two sets of numbers. The first set uses the standard algorithms, and the other is an example of the time-space tradeoff mentioned previously.

Digital Signature Schemes

A digital signature is used to attach an unalterable tag to an electronic document to show that the signer approved the document. Here, we describe how the interactive identification schemes discussed previously can be transformed into digital signature schemes. We also compare the performance and levels of provable security of these and other signature schemes.

The digital signature schemes we describe "sign" (i.e., attach a signature to), at most, a specific number of bits of the message at one time. If the document or message is too long (i.e., exceeds this limit), then one could simply separate the message into blocks of the proper size and sign the individual blocks.

This has the disadvantage that the blocks now are independent. Unless a scheme exists to identify the proper order of the blocks, anyone could reassemble some of the block signatures in a different order and, thus, sign any message that could be constructed by reordering or deleting some of the blocks. Another disadvantage is that a prohibitive number of computations may be needed to sign a very long message.

One solution is to apply a hash function h to the message, to produce a block of the proper size that can then be signed. *Hash function* is a term commonly used in computer science to denote a function from a large set into a small set.

For cryptographic applications, we need hash functions that satisfy the following criteria:

- The function h should be easy to compute.
- It should be difficult to find two distinct values x_1 and x_2 such that $h(x_1) = h(x_2)$. A hash function that satisfies this is called *collision free.*
- For a given value y, it should be difficult to find a value x such that $h(x) = y$. Such a hash function is called *one way.*

Hash functions that satisfy these properties are called *cryptographically secure hash functions.*

There are no known techniques for proving that a given function satisfies these properties. However, several functions have been shown to satisfy these properties if certain computational problems are infeasible,[18] or if there are provably good block cyphers.[19,20] Other functions that are even more computationally efficient have been suggested in the open literature as cryptographically secure hash functions; for example, the MD4[21] and Snefru[22] hash functions.

Converting Identification Schemes. In the interactive identification schemes discussed earlier, the verifier's only actions were to receive an initial transmission from the prover, and to choose a random string of bits that she sent to the prover. To convert the identification scheme into a signature scheme, we simply use a cryptographically secure hash function—which is applied to the initial transmission and the message—to generate the random string of bits. Specifically, for our scheme, the signature version would be:

Table II. Efficiency of Signature Schemes

	Schnorr		Sandia		Fiat–Shamir		Micali–Shamir	RSA	ElGamal		NIST (DSS)	
Identity based?	no		no		yes	no	no	no	no		no	
Provable witness hiding?	no		yes		no	yes	yes	no	no		no	
Length of public key (bytes)	64		64		8192	8192	70	64	64		64	
Length of secret key (bytes)	18		64		8192	8192	8192	64	64		20	
Signer's off-line multiplications (number of operations)	207	47*	765	129*	1	1	1	0	765	129*	237	52*
Signer's auxiliary storage (bytes)	146	2450	128	6720	64	64	0	NA	128	6720	148	2772
Signers's on-line multiplications (number of operations)	< 1		< 1		64	64	64	600†–765	1		< 1	
Length of signature (bytes)	34		80		80	80	80	64	128		40	
Multiplications by the verifier (number of operations)	239	211*	797	293*	65	65	3	2	893	689*	277	229*
Verifier's auxiliary storage (bytes)	128	2450	128	6720	NA	NA	NA	NA	128	8320	148	2176

NA = not applicable.
* This value reflects the use of precomputed values (see Panel 2).
† In some instances, the Chinese remainder theorem can be used to generate an RSA signature in the equivalent of about 150 multiplications.

1. When the user wants to sign a message, m, he chooses a random number $r \in \{1, \cdots, p-1\}$, and computes $x \equiv \alpha^r \pmod{p}$. Then, he computes $e = h(x, m)$ and an integer y such that $y \equiv r + Se \pmod{p-1}$. The pair (e, y) is this user's signature for m.
2. To verify a signature, a verifier first checks the authenticity of the signer's public key by verifying the signature $S(I, K)$. Then, the verifier computes $x \equiv \alpha^y K^e \pmod{p}$. She checks that $h(x, m) = e$. If this condition is satisfied, she accepts (e, y) as the signer's signature of m.

The other interactive identification schemes discussed earlier can be similarly modified.

For the signature variations of the schemes, we must use a longer e than we did for interaction.

Panel 3. A New Signature Standard

On August 30, 1991, the National Institute of Standards and Technology (NIST) announced a proposed standard for producing digital signatures.[25] This standard, which is a variation of the ElGamal signature scheme,[24] introduces a few innovations that reduce the amount of computation required and the size of digital signatures.

Here, we provide a brief description of the scheme. (For instructions about how to obtain a complete description of the proposed standard, see Reference 25.)

Every person who uses the scheme will require the following parameters:

- A prime q with $2^{159} < q < 2^{160}$
- A prime p with $2^{511} < p < 2^{512}$ and $p \equiv 1 \pmod{q}$
- A number g with $g^q \equiv 1 \pmod{p}$ and $1 < g < p$.

In addition to these values, each user will require some personal parameters:

- A secret number S with $1 < S < q$
- A public number K with $K \equiv g^S \pmod{p}$.

The proposed standard requires two components that are not specified as part of the standard:

- A method for generating random integers
- A hash function H that transforms a message m into a 160-bit hash value $H(m)$.

To generate a signature for a message m, a user first chooses a random number r with $1 < r < q$. Then, the user computes:

$$x = (g^r \bmod p) \bmod q$$

$$y = r^{-1}\,[H(m) + Sx] \bmod q.$$

If $0 < x < q$ and $0 < y < q$, then the pair (x, y) constitutes the signature of the message m. (Otherwise, the signer should choose a new r.)

To verify the signature, the recipient would first verify the authenticity of the signer's public key K (see the text for a description of K). The recipient would then check to see that $0 < x < q$ and $0 < y < q$, and reject the signature if either condition fails to hold. Next, the recipient computes:

$$w = y^{-1} \bmod q$$

$$u_1 = wH(m) \bmod q$$

$$u_2 = wx \bmod q$$

$$v = (g^{u_1} K^{u_2} \bmod p) \bmod q.$$

If $v = x$, then the recipient can assume that the signature is legitimate.

Otherwise, an attacker could use an off-line attack to generate a false signature. (Because the identification schemes were interactive, an impostor had to respond in real time.) A natural choice for the size of e is 128 bits, because the number of choices for e needs to be big enough that an exhaustive search is infeasible. The MD4 and Snefru hash functions produce outputs of this size.

Comparison of Signature Schemes. In Table II, we compare the computation, communications, storage, and levels of provable security for different signature schemes. We consider schemes derived from the interactive identification schemes mentioned earlier, as well as the RSA[23] and ElGamal[24] signature schemes. (For complete descriptions of the latter schemes, we refer interested readers to the original sources.) In addition, we include data for the Digital Signature Standard, abbreviated as DSS,[25] that was recently announced by the National Institute of Standards and Technology (NIST). Panel 3 describes this new standard.

To prove the security of any of these schemes, we require an assumption about the hash function that is used. With this additional assumption, the provable security of the interactive identification schemes carries over to their signature variations.

The provable security of the RSA, ElGamal, and DSS schemes is too complicated to describe in detail here. Briefly, the security of the RSA scheme relates to the problem of factoring, while the security of the ElGamal and DSS schemes relates to the problem of computing discrete logarithms. The only known approach to breaking these three systems requires that the problem of factoring be solved or that discrete logarithms be computed. However, there is no proof that breaking any of these three schemes requires the solution of these problems.

While the quantitative performance of some of these signature schemes is closely related to the performance of the related identification schemes, there are some subtle differences. For example, the verifier's computations are always done in real time for an identification scheme, but this need not be done in a signature scheme if signatures are generated more often than they are verified. In addition, the specific application for a signature scheme may dictate that signatures be verified more often than they are generated. Therefore, comparisons between schemes may depend heavily on the scenarios in which they are to be used.

Table II compares the efficiency of various signature schemes for the following choices of parameters:

- Schnorr: p has 512 bits, q has 140 bits, e has 128 bits.
- Sandia: p has 512 bits, q has 140 bits, e has 128 bits.
- Fiat–Shamir: n has 512 bits, $k = 128$, $t = 1$.
- Micali–Shamir: n has 512 bits, $k = 128$, $t = 1$.
- RSA: n has 512 bits.
- ElGamal: p has 512 bits.
- NIST (DSS): p has 512 bits, q has 160 bits.

Once again, the communications and storage figures are given in bytes. Also, we give multiple operation counts for the Schnorr, Sandia, ElGamal, and NIST schemes for the two different approaches to exponentiation.

For the RSA scheme, estimates of the number of on-line multiplications required by the signer depend on whether he used an addition chain or the basic method of exponentiation.

Conclusion

From Tables I and II, none of the identification or signature schemes is clearly the best in all situations. Also, the Sandia scheme's performance is highly competitive with the other schemes, even though its security is based on the difficulty of solving two hard problems.

The Sandia scheme can be easily implemented in current technology. An individual could use a hand-held device that stores the system's private keys in encrypted form. To use the device, he would enter a secret password, which would be used to decrypt the private keys. Then, the protocol could be started.

For most applications, it would be reasonable to implement the scheme on a *smart card.* (This device is the size of a credit card and has an embedded microprocessor.) Even for a smart card that has minimal computational power, the off-line computation would take only a few seconds, and the on-line computation would take much less than a second.

Thus, it is possible to use current technology to design a system that can address all the problems with identification and signatures mentioned in this paper's introduction. Furthermore, this system does not have many of the disadvantages associated with the use of passwords. We expect that systems such as this will soon find their way into the marketplace.

Acknowledgments. The authors wish to thank Russell Maxwell, James Holmes, John Eldridge, and Lyndon Pearson of Sandia National Laboratories for helpful conversations during the writing of this paper.

References

1. W. Diffie and M. Hellman. "New directions in cryptography," *IEEE Transactions on Information Theory*, Vol. 22, No. 6, November 1976, pp. 472–492.
2. J. G. Steiner, C. Neuman, and J. I. Schiller, "Kerberos: An authentication service for open network systems," *Proceedings of the USENIX Winter Conference*, Dallas, Texas, February 9–12, 1988, The USENIX Association, Berkeley, California, February 1988, pp. 191–202.
3. R. M. Needham and M. D. Schroeder, "Using encryption for

authentication in large networks of computers," *Communications of the ACM*, Vol. 21, November 1985, pp. 993–999.
4. S. M. Bellovin and M. Merritt, "Limitations of the kerberos authentication system," *Computer Communication Review*, Vol. 20, No. 5, October 1990, pp. 119–132.
5. J. P. Holmes, R. L. Maxwell, and L. J. Wright, "A performance evaluation of biometric identification devices," Technical Report No. SAND91–0276, Sandia National Laboratories, Albuquerque, New Mexico, July 1990.
6. W. R. Cheswick. "Design of a secure internet gateway," *Proceedings of the Summer 1990 USENIX Conference*, Anaheim, California, June 11–15, 1990, The USENIX Association, Berkeley, California, June 1990, pp. 233–237.
7. U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," *Proceedings of the 22nd ACM Symposium on Theory of Computing*, Baltimore, Maryland, May 14–16, 1990, Association for Computing Machinery, New York, 1990, pp. 416–424.
8. A. Fiat and A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology (Proceedings of Crypto '86), Lecture Notes in Computer Science*, Vol. 263, Springer-Verlag, New York, 1987, pp. 186–199.
9. A. K. Lenstra and M. S. Manasse, "Factoring by electronic mail," *Advances in Cryptology—Proceedings of Eurocrypt '89, Lecture Notes in Computer Science*, Vol. 434, Springer-Verlag, New York, 1990, pp. 355–371.
10. C. Pomerance, *Factoring, Proceedings of Symposia in Applied Mathematics*, Vol. 42, American Mathematical Society, Providence, Rhode Island, 1990, pp. 27–48.
11. B. LaMacchia and A. Odlyzko, "Computation of discrete logarithms in prime finite fields," *Advances in Cryptology—Proceedings of Crypto '90, Lecture Notes in Computer Science*, Vol. 537, A. Menezes and S. A. Vanstone (eds.), Springer-Verlag, New York, to be published in November 1991.
12. K. S. McCurley, "The Discrete Logarithm Problem," *Proceedings of Symposia in Applied Mathematics*, Vol. 42, American Mathematical Society, Providence, Rhode Island, 1990, pp. 49–74.
13. E. F. Brickell and K. S. McCurley. "An interactive identification scheme based on discrete logarithms and factoring," *Journal of Cryptology*, to be published in Vol. 5, No. 1, 1992.
14. C. P. Schnorr, "Efficient identification and signatures for smart cards," *Advances in Cryptology—Proceedings of Crypto '89, Lecture Notes in Computer Science*, Vol. 435, Springer-Verlag, New York, 1990, pp. 239–252.
15. U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identify," *Journal of Cryptology*, Vol. 1, No. 2, 1988, pp. 77–94.
16. S. Micali and A. Shamir, "An improvement of the Fiat–Shamir identification and signature scheme," *Advances in Cryptology (Proceedings of Crypto '88), Lecture Notes in Computer Science*, Vol. 403, Springer-Verlag, New York, 1990, pp. 244–247.
17. D. E. Knuth, *Seminumerical Algorithms, The Art of Computer Programming*, Vol. 2, Second edition, Addison-Wesley, Reading, Massachusetts, 1981, pp. 441–505.
18. I. B. Damgärd, "Collision free hash functions and public key signature schemes," *Advances in Cryptology—Proceedings of Eurocrypt '87, Lecture Notes in Computer Science*, Vol. 304, Springer-Verlag, New York, 1988, pp. 203–216.
19. R. C. Merkle, "One way hash functions and DES," *Advances in Cryptology—Proceedings of Crypto '89, Lecture Notes in Computer Science*, Vol. 435, Springer-Verlag, New York, 1990, pp. 428–446.
20. J.-J. Quisquater and M. Girault, "2*n*-bit hash-functions using *n*-bit symmetric block cipher algorithms," *Advances in Cryptology—Proceedings of Eurocrypt '89, Lecture Notes in Computer Science*, Vol. 434, Springer-Verlag, New York, 1990, pp. 102–109.
21. R. L. Rivest, "The MD4 message digest algorithm," *Advances in Cryptology—Proceedings of Crypto '90, Lecture Notes in Computer Science*, Vol. 537, A. Menezes and S. A. Vanstone (eds.), Springer-Verlag, New York, to be published in November 1991.
22. R. C. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 43–58.
23. R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, February 1978, pp. 120–126.
24. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, Vol. 31, No. 4, July 1985, pp. 469–472.
25. "A Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)," *Federal Register*, Vol. 56, No. 169, August 31, 1991, pp. 42980–42982.
26. E. F. Brickell, D. M. Gordon, and K. S. McCurley, "Fast Exponentiation with Precomputation," Technical Report No. SAND91–1836C, Sandia National Laboratories, Albuquerque, New Mexico, October 1991.

Biographies (continued)

and has a B.S. in mathematics from the University of Santa Clara (California), and both an M.S. in statistics and a Ph.D. in mathematics from the University of Illinois (Urbana).

(Manuscript received May 13, 1991)