Ernest F. Brickell (Ed.)

CLOA-740

Advances in Cryptology – CRYPTO '92

12th Annual International Cryptology Conference Santa Barbara, California, USA August 16-20, 1992 Proceedings

Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo Hong Kong Barcelona Budapest Series Editors

Gerhard Goos Universität Karlsruhe Postfach 6980 Vincenz-Priessnitz-Straße 1 D-76131 Karlsruhe, Germany Juris Hartmanis Cornell University Department of Computer Science 4130 Upson Hall Ithaca, NY 14853, USA

Volume Editor

Emest F. Brickell Department 1423, Sandia National Laboratories PO Box 5800, Albuquerque, NM 87185, USA

5347

CR Subject Classification (1991): E.3-4, D.4.6, G.2.1

ISBN 3-540-57340-2 Springer-Verlag Berlin Heidelberg New York ISBN 0-387-57340-2 Springer-Verlag New York Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1993 Printed in Germany

Typesetting: Camera-ready by author Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr. 45/3140-543210 - Printed on acid-free paper

Preface

Crypto'92 took place on August 16-20, 1992. It was the twelfth in the series of annual cryptology conferences held on the beautiful campus of the University of California, Santa Barbara. Once again, it was sponsored by the International Association for Cryptologic Research, in cooperation with the IEEE Computer Society Technical Committee on Security and Privacy. The conference ran smoothly, due to the diligent efforts of the general chair, Spyros Magliveras of the University of Nebraska.

One of the measures of the success of this series of conferences is represented by the ever increasing number of papers submitted. This year, there were 135 submissions to the conference, which represents a new record. Following the practice of recent program committees, the papers received anonymous review. The program committee accepted 38 papers for presentation. In addition, there were two invited presentations, one by Miles Smid on the Digital Signature Standard, and one by Mike Fellows on presenting the concepts of cryptology to elementary-age students. These proceedings contains these 40 papers plus 3 papers that were presented at the Rump Session. I would like to thank all of the authors of the submitted papers and all of the speakers who presented papers.

I would like to express my sincere appreciation to the work of the program committee: Ivan Damgard (Aarhus University, Denmark), Oded Goldreich (Technion, Israel), Burt Kaliski (RSA Data Security, USA), Joe Kilian (NEC, USA), Neal Koblitz (University of Washington, USA), Ueli Maurer (ETH, Switzerland), Chris Mitchell (Royal Holloway, UK), Kazuo Ohta (NTT, Japan), Steven Rudich (Carnegie Mellon, USA), and Yacov Yacobi (Bellcore, USA). I would also like to thank Joan Boyar for agreeing to chair one of the sessions.

Ernest Brickell Albuquerque, NM August, 1993

BIBLIOTHEQUE DU CERIST

- - - - - -

PROCEEDINGS OF CRYPTO '92

TABLE OF CONTENTS

Session I: Digital Signatures and Identification I	
Provably Unforgeable Signatures Jurjen N. E. Bos, David Chaum	1
New Constructions of Fail-Stop Signatures and Lower Bounds Eugéne van Heijst, Torben P. Pedersen, Birgit Pfitzmann	15
Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes Tatsuaki Okamoto	31
An Efficient Digital Signature Scheme Based on an Elliptic Curve Over the Ring Z _n Tatsuaki Okamoto, Atsushi Fujioka, Eiichiro Fujisaki	54
Session II: The Digital Signature Standard	
Designing and Detecting Trapdoors for Discrete Log Cryptosystems Daniel M. Gordon	66
Response to Comments on the NIST Proposed Digital Signature Standard (Invited) Miles E. Smid, Dennis K. Branstad	76
Session III: Applications and New Problems	
Wallet Databases with Observers David Chaum, Torben Pryds Pedersen	89
Making Electronic Refunds Safer Rafael Hirschfeld	106
Fair Public-Key Cryptosystems Silvio Micali	113
Pricing via Processing or Combatting Junk Mail Cynthia Dwork, Moni Naor	139
Session IV: Secret Sharing I	
On the Information Rate of Secret Sharing Schemes Carlo Blundo, Alfredo De Santis, L. Gargano, Ugo Vaccaro	148

New General Lower Bounds on the Information Rate of Secret Sharing Schemes Doug R. Stinson	168
Universally Ideal Secret Sharing Schemes Amos Beimel, Benny Chor	183
Session V: Theory I	
Perfect Zero-Knowledge Arguments for NP can be Based on General Complexity Assumptions	196
Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, Moti Yung	
Low communication 2-Prover Zero-Knowledge Proofs for NP Cynthia Dwork, Uri Feige, Joe Kilian, Moni Naor, Muli Safra	215
Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent Shafi Goldwasser, Rafail Ostrovsky	228
On the Discrepancy between Serial and Parallel of Zero-Knowledge Protocols Kouichi Sakurai, Toshiya Itoh	245
Session VI: Cryptographic Functions	
On the Design of SP Networks form an Information Theoretic Point of View M. Sivabalan, Stafford Tavares, L. E. Peppard	260
Partially-Bent Functions Claude Carlet	280
Session VII: Digital Signatures and Identification II	
Practical Approaches to Attaining Security Against Adaptively Chosen Ciphertext Attacks Yuliang Zheng, Jennifer Seberry	292
On the Security of the Permuted Kernel Identification Scheme Thierry Baritaud, Mireille Campana, Pascal Chauvaud, Henri Gilbert	305
Session VIII: Computational Number Theory	
Massively Parallel Computation of Discrete Logarithms Daniel M. Gordon, Kevin S. McCurley	312
A Quadratic Sieve on the <i>n</i> -Dimensional Cube René Peralta	324
Efficient Multiplication on Certain Nonsupersingular Elliptic Curves Willi Meier, Othmar Staffelbach	333

Speeding up Elliptic Cryptosystems by Using a Signed Binary Window Method Kenji Koyama, Yukio Tsuruoka	345
On Generation of Probable Primes by Incremental Search Jørgen Brandt, Ivan Damgård	358
Session IX: Cryptography Education	
Kid Krypto (Invited) Michael Fellows, Neal Koblitz	371
Session X: Theory II	
On Defining Proofs of Knowledge Mihir Bellare, Oded Goldreich	390
Public Randomness in Cryptography Amir Herzberg, Michael Luby	421
Necessary and Sufficient Conditions for Collision-Free Hashing Alexander Y. Russell	433
Certifying Cryptographic Tools: The Case of Trapdoor Permutations Mihir Beilare, Moti Yung	442
Session XI: Key Distribution	
Protocols for Secret Key Agreement by Public Discussion Based on Common Information Ueli M. Maurer	461
Perfectly-Secure Key Distribution for Dynamic Conferences Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, Moti Yung	471
Session XII: DES	
Differential Cryptanalysis of the Full 16-Round DES Eli Biham, Adi Shamir	487
Iterative Characteristics of DES and S ² DES Lars R. Knudsen	497
DES is not a Group Keith W. Campbell, Michael J. Wiener	512
A High-Speed DES Implementation for Network Applications Hans Eberle	521

Session XIII: Secret Sharing JI

Threshold Schemes with Disenrollment Bob Blakley, G. R. Blakley, Agnes H. Chan, James L. Massey	540
Non-Existence of Homomorphic General Sharing Schemes for Some Key Spaces Yair Frankel, Yvo Desmedt, Mike Burmester	549
An I-Span Generalized Secret Sharing Scheme Lein Harn, Hung-Yu Lin	558
Session XIV: Rump Session	
Provable Security Against Differential Cryptanalysis Kaisa Nyberg, Lars R. Knudsen	566
Content-Addressable Search Engines and DES-like Systems Peter C. Wayner	57 5
FFT-Hash-II is not yet Collision-free Serge Vaudenay	587

Provably Unforgeable Signatures

Jurjen N.E. Bos* David Chaum[†]

Abstract. Very strong definitions of security for signature schemes have been proposed in the literature. Constructions for such schemes have been proposed, but so far they have only been of theoretical interest and have been considered far too inefficient for practical use.

Here we present a new scheme that satisfies these strongest definitions and uses essentially the same amount of computation and memory as the widely applied RSA scheme. The scheme is based on the well known RSA assumption.

Our signatures can be thought of as products resulting from a two-dimensional Lamport scheme, where one dimension consists of a list of public constants, and the other is the sequence of odd primes.

Introduction

One of the greatest achievements of modern cryptography is the digital signature. A digital signature on a message is a special encryption of the message that can easily be verified by third parties. Signatures cannot be denied by the signer nor falsified by other parties.

This article introduces a new signature scheme that combines the strength of the strongest schemes with the efficiency of RSA.

Signing a message of 245 bits in our scheme is possible in roughly 910 multiplications, and verifying it costs about 152 multiplications. In comparison, RSA, using the ISO/IEC standard 9796 redundancy scheme, takes roughly 768 multiplications (or 610 using addition chains) for signing, and 3 (or optionally 17) for verification. RSA signatures are 512 bits long, while ours requires an additional message counter. Thus, 16 extra bits give a scheme that allows 65,536 signatures per public key.

A variation involving pre-computation, signs short messages (64 bits) in 33 multiplications (not counting precomputation) and verifies in 35 multiplications.

After the introduction, we discuss other signature schemes relevant to this work. We discuss the Lamport signature scheme, on which this signature scheme is based, in detail. Then, the new scheme is explained, and the possible choices for parameter valucs are shown.

^{*} This article is adapted from the dissertation "Practical Privacy" of Jurjen N.E. Bos, written while he was at CWI (the Dutch nationally funded centre for Mathematics and Computer Science). He is currently affiliated with Irdeto (a pay TV company) in Hoofddorp, Netherlands.

[†] David Chaum is affiliated both with CWI and DigiCash (innovators in electronic money systems).

Signature scheme

An overview of signature schemes, comparing securities, can be found in the paper mentioned earlier [GMR88]. We use their notation. They define a signature scheme as consisting of the following components:

- A security parameter k, that defines the security of the system, and that may also influence performance figures such as the length of signatures, running times and so on.
- A message space M, that defines on which messages the signature algorithm may be applied.
- A signature bound b, that defines the maximal number of signatures that can be generated without reinitialization. Typically, this value depends on k, but it can be infinite.
- A key generation algorithm G, that allows a user to generate a pair of corresponding public and secret keys for signing. The secret key S is used for generating a signature, while the public key P is used to verify the signature.
- A signature algorithm σ , that produces a signature, given the secret key and the message to be signed.
- finally, a verification algorithm, that produces true or false on input of a signature and a public key. It ouputs true if and only if the signature is valid for the particular public key.

Some of these algorithms may be randomized, which means that they may use random numbers. Of course, G must be randomized, because different users must produce different signatures. The signing algorithm σ is sometimes randomized, but this tends to produce larger signatures. The verification algorithm is usually not randomized.

A simple example of a signature scheme is a trapdoor one-way function f. The function f is used for verification by comparing the function value of the signature with the message to be signed, and σ is the trapdoor of f. The main problem with such a scheme is that random messages f(x) can be signed by taking a random signature value x. A simple solution is to let M be a sparse subset of a larger space, so that the probability that f(x) is a valid message for random x is low. An example of a sparse subset is the set of "meaningful" messages.

Related work

The notion "digital signature" was introduced in [DH76]. This paper, which can be considered the foundation of modern cryptography, discusses the possibility of digital signatures and the use of a trapdoor one-way function to make them.

[RSA78] is the original article on the RSA scheme. It introduces the famous RSA trapdoor one-way function. This function is still widely in use and is applied frequently. A well-known weakness of RSA is that it is multiplicative: the product of two signatures is the signature of the product. This potential problem can be prevented as above by choosing an appropriate sparse message space.

Since then, an enormous number of signature schemes have been proposed [Rab77, MH78, Sha78, Rab79, Lie81, DLM82, GMY83, Den84, GMR84, OSS84, EIG85, OS85, FS86, BM88, GMR88, CA89, EGL89, EGM89, Mer89, Sch89, SQV89, BCDP90, Cha90, CR90, Hay90, CHP91], applied [Wil80, Cha82, Gol86, Bet88], and broken [Yuv79, Sha82, Tu84, BD85, EAKMM85, Roo91]. We will not discuss all these schemes here; we only discuss the ones that are interesting to compare with the new scheme.

The schemes [Rab79, GMY83, GMR84, GMR88] are steps towards a provably secure signature scheme. The scheme described in the last article is secure in a very strong way: it is "existentially unforgeable under an adaptive chosen-message attack" with probability smaller than 1/Q(k) for every polynomial Q. This means that generating a new signature is polynomially hard if signatures on old messages are known, even if the old signatures are on messages chosen by the attacker.

The scheme in [GMR88] is based on factoring. While our scheme is based on the slightly stronger RSA assumption, it is much more efficient. The signature scheme of [GMR88] uses a large amount of memory for the signer, and quite a lot of computation. Our scheme uses no memory at all, except for a counter and the public values, and signing and verifying takes about as much computation as RSA does, depending on the parameters.

The Lamport Scheme

Our scheme can be thought of an optimization for both security and efficiency of [GMY83]. To explain the new system, we compare it to the earlier *Lamport scheme* (explained already in [DH76, page 650]). To make a signature in this scheme, the signer makes a secret list of 2k random numbers

 $\mathbf{A} = a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \dots, a_{k,0}, a_{k,1},$

applies a one-way function f to all elements, and publishes the result **B**:

$$\mathbf{B} = \begin{cases} f(a_{1,0}), f(a_{2,0}), \dots, f(a_{k,0}) \\ f(a_{1,1}), f(a_{2,1}), \dots, f(a_{k,1}) \end{cases}$$

The signature consists of the numbers $a_{1,m_1}, a_{2,m_2}, \ldots, a_{k,m_k}$ from the list A (one from each "column"), where m_1, m_2, \ldots, m_k are the bits of the message to be signed. The lists A and B cannot be used again.

The properties of Lamport's scheme are easy to verify:

- · Signing a message is only the publication of the proper elements of A.
- To forge a signature, one needs to find certain values from the list A. How hard this is, depends on the security of the one-way function f.

4

- If the values A are only used for one signature, new signatures cannot be made from old ones.
- Verification of a signature consists of applying the one-way function to the signature values, and comparing them to the public values determined by the signed message.

The new system uses the same idea, with three important differences. first, the list B is replaced by another list that can be used for all signatures. Second, the list A is constructed from two lists so that less memory is needed to define it. Third, the elements of A in the signature can be combined into a single number.

A small optimization

There is a trivial optimization of Lamport's scheme that reduces the number of public function values to almost half, that we could not find in the literature. This optimization is independent of the signature scheme as such. Basically, the signer signs by publishing a k-element subset of the 2k secret numbers. Lamport's scheme chooses a particular set of subsets of the set of 2k elements, as shown above. The necessary property of this set of subsets is that no subset includes another.

There are other sets of subsets with the property that no subsets includes another. A largest set of subsets with this property is the set of all k-element subsets (a wellknown result from lattice theory). For these sets, it is easy to see that no subset includes another.

For example, in Lamport's scheme, the list of 6 elements

 $\mathbf{A} = a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, a_{3,0}, a_{3,1}$

allows us to sign messages of 3 bits. If we renumber A as a_1,a_2,a_3,a_4,a_5,a_6 , we get the set of 20 three-element subsets of A:

 $\begin{array}{l} (a_1,a_2,a_3), \{a_1,a_2,a_6\}, \{a_1,a_2,a_5\}, \{a_1,a_2,a_6\}, \{a_1,a_3,a_4\}, \\ (a_1,a_3,a_5), (a_1,a_3,a_6\}, \{a_1,a_4,a_5\}, \{a_1,a_4,a_6\}, \{a_1,a_5,a_6\}, \\ \{a_2,a_3,a_4\}, \{a_2,a_3,a_5\}, \{a_2,a_3,a_6\}, \{a_2,a_4,a_5\}, \{a_2,a_4,a_6\}, \\ \{a_2,a_5,a_6\}, \{a_3,a_4,a_5\}, \{a_3,a_4,a_6\}, \{a_3,a_5,a_6\}, \{a_4,a_5,a_6\}; \\ \end{array}$

this allows us to sign one of 20 messages, which is equivalent to more than 4 bits.

In general, there are

$$\binom{2k}{k}$$
, or about $\frac{2^{2k}}{\sqrt{k\pi}}$,

k-element subsets, so that we can sign messages of about $2k - \frac{1}{2}\log_2(k\pi)$ bits. The original Lamport scheme allowed messages of only k bits, so that we get almost **BIBLIOTHEQUE DU CERIST**

a doubling of the message size for the same size of the list **B**. This simple improvement can also be used in our new signature scheme.

To encode a signature, a mapping needs to be defined between messages and these subsets:

The simplest mapping just enumerates messages (interpreted as numbers from 0 onwards) to sets (seen as binary strings that denote 1 for presence and 0 for absence) in order. Such a mapping is easily and efficiently computed by the algorithm shown in figure 1. The binomial coefficients do not need to be computed by repeated multiplication and division. The first binomial coefficient is always the same, so it can be precomputed, and the others can be computed by one multiplication and one division by small numbers using the properties:

$$\binom{t-1}{e} = \binom{t}{e} \cdot \frac{t-e}{t} \text{ and } \binom{t-1}{e-1} = \binom{t}{e} \cdot \frac{e}{t}.$$

The algorithm outputs ones and zeros corresponding to the elements in the resulting set.

Note that the Lamport scheme uses another mapping that maps numbers onto kelement subsets, but that only a small number of these sets are used.

Let *n*, the message, be a number in the range
$$0...\binom{2k}{k}-1$$
.
Put 2k in t and k in e.
While $t > 0$:
Put $t-1$ in t.
If $n \ge \binom{t}{e}$, put $n - \binom{t}{e}$ in n, $e-1$ in e, and output a 1 (this t is in the set).
Else, output a 0 (this t is not in the set).

Fig 1. Algorithm for the mapping s.

The New Signature Scheme

The new signature scheme replaces the list A of the Lamport scheme by a list of numbers that can be organized in a matrix. Instead of using a new list B for every signature, a fixed list called **R** is used for all signatures and all participants. The one-way function f is replaced by a set of trapdoor one-way functions, that changes per signature. For the trapdoor one-way functions, we use the modular root function of [RSA78].

The construction allows us to sign long messages using only a few numbers to define the set A. In the example of figure 2, the set A of 12 elements is constructed from three primes p_1 , p_2 , p_3 (used only for this signature) and four public values r_1 , r_2 , r_3 , r_4 (that can be used again). This set allows us to sign messages of 9 bits, since there are $924 > 2^9$ possible 6-element subsets of A. Signing messages of 9 bits in the original Lamport scheme takes 18 public values that can be used only once.

$\sqrt[n]{r_1}$	$\sqrt[p_1]{r_2}$	PV73	$\sqrt[p_1]{r_4}$
$P_{2}\sqrt{r_{1}}$	P2 12	7/3	$\sqrt[P_2]{r_4}$
<i>¶</i> √7	$\sqrt[n]{r_2}$	P3 [r3	$\sqrt{r_4}$

Fig. 2. Example list A of the new scheme.

The numbers α of A are secret encryptions of the numbers r_i of R, and the corresponding decryption exponents are public. The multiplicative property of RSA allows us to multiply the values of the signature to form one number. Verification of a signature can be done using a simple computation, without having to compute the separate factors.

The public values of the new system are:

- One modulus per signer;
- The system-wide list R. This list is used by all users, and that it does not change often, so that distribution does not require much traffic. The numbers in R are smaller than the smallest modulus used by the signers.
- A list of sets of primes that may be used for signing. For security reasons, the sets may not overlap each other, and the signers may only use these sets of primes.

A signature consists of the original message signed, the signature proper (an integer smaller than the modulus of the signer), and a description of the prime set.

In the language of [GMR88]:

- The security parameter determines the size of the RSA modulus. This modulus can vary per user.
- The message space M is (equivalent to) the set of subsets of A that include half the elements.
- The size of the public list of sets of primes determines the signature bound b.
- Key generation is a matter of generating an RSA modulus, and computing exponents for the modular root extractions.
- Signing and verification are defined below.

Signing

For the list A of a signature, the set of RSA encryptions

$$\mathbf{A} = \left\{ \sqrt[p]{r} \mod n \,\middle|\, p \in \mathbf{P}; r \in \mathbf{R} \right\}$$

is used, where:

- P a set of primes from the public list;
- R is the public list of verification values;
- n is the RSA modulus of the signer.

As explained above, a signature is constructed from a subset determined by s(m) of half these numbers. The constant k used in the algorithm that maps s is equal to $\left\lfloor \frac{\#P \#R}{2} \right\rfloor$. This allows us to sign a message of almost #A = #P #R bits. The product of the elements of A in this subset is the signature. Since this is a single number, the signature is much more compact than in Lamport's scheme.

Thus, signing a message consist of the following steps:

• Choose the set P of primes that is to be used for this signing from the public list. This determines A:

$$\mathbf{A} = \left\{ \sqrt[p_j]{r_j} \mod n \, \middle| \, i, j \in \{1, \dots, \# \mathbf{P}\} \times \{1, \dots, \# \mathbf{R}\} \right\}.$$

Like the sets A and B in Lamport's scheme, the set P can be used only once. The list A need not be computed.

- Determine the message *m* to sign. This could be a message, or a public hash function value of that message, for example.
- Compute the subset M of index pairs from {1,...,#P}×{1,...,#R} from the message m with the algorithm described above:

 $\mathbf{M} = s(m)$

Compute the signature proper:

$$S = \prod_{i,j \in \mathbf{M}} {}^{p_i} \sqrt{r_j} \pmod{n},$$

and send m, P, and S to the recipient.

There are two ways to increase the efficiency of signing. If there is time to do a precomputation, the entire set A can be computed before the value of m is known. Although this takes quite a while, signing becomes much faster, since signing consists only of multiplying the proper values of A together. If precomputation is not possible, the computation of S can be speeded up with a vector addition chain [Bos92].

Verification

Instead of trying to compute individual factors of the signature, the number S can be verified in a single computation. To see this, we note that the power of the signature

should be equal to the following product that can be computed from public values:

 $\prod_{i,j\in\mathbf{M}} r_i^{\lim_{k\in\mathbf{F}} p_k/p_j}$

The lower product can be computed with a vector addition chain. Verification of a signature consists of checking that these two values are the same. The verification can be performed with a single vector addition chain, if the inverse of the signature is computed first:

$$(S^{-1})^{k \in \mathbb{P}} \cdot \prod_{i,j \in M} r_i^{k \in \mathbb{P}} / r_j$$

which must evaluate to $1 \pmod{n}$. To increase the efficiency of the verification, the signer could send 1/S instead of S, so that the inversion is performed only once by the signer, and not by every verifier.

If not all prime numbers from P occur as exponents in the set M, it is possible to verify a signature using slightly fewer multiplications by raising S to only the occurring primes. Unfortunately, this optimization is only applicable in the less interesting cases where verification requires a lot of multiplications.

The verifier must also check whether P occurs in the public list. If P is described as an index number in this list, this is of course unnecessary.

Parameters

In practice, the following parameter values could be used:

- A modulus size big enough to make factorization hard (200 digits, or 668 bits).
- R a list of 50 numbers.
- The sets **P** consisting of the $(5n+1)^{th}$ to the $(5n+5)^{th}$ odd prime number, where $n \in \{0, \dots, 16404\}$ is the sequence number of the signature. This uses the primes of up to 20 bits.

With these parameters, we have sets A of 250 elements, so that a message of 245 bits (30 bytes) can be signed. A signature consists of the message, the signature product (668 bits, or 84 bytes), and the index number of the prime set (15 bits, or 2 bytes). Computing a signature takes about 1512 modular multiplications, and verification about 272; both these numbers are obtained using vector addition chains.

The list of the odd primes up to 20 bits (the highest being 1048557) can easily be stored; it would need only 64 K bytes of storage (using a bit table of the odd numbers) and contain 82025 primes. Such a list can easily be stored in a ROM chip. When all primes are used up, the user can choose a new modulus and start again. Another solu-

∏.^pk S^{keP} tion is to change the list \mathbf{R} often enough so that users do not run out of primes. To make it possible to verify old signatures, old values of R and the user moduli must be saved.

The list R can be computed from a seed number using a public hash function. This way, only one seed number is needed to define R. This allows us one to use a long list R while using small amounts of data to distribute it. Also, less data is needed to save old lists.

Figure 3 shows the performance of the algorithm for several sizes of \mathbf{R} and \mathbf{P} . For each of the entries in the table, the modulus is 668 bits (200 decimal digits), and the size of the primes in **P** is 20 bits. The entries are computed by averaging random number approximations. The entries marked by * have an estimated standard deviation higher than 10, so that the last digits are likely to be inaccurate.

Powers and products were computed using addition chains and sequences; see [Bos92, chapter 4]. The products were computed collecting the base numbers; for example, the product

 $b_2^{e_1} \cdot b_3^{e_1} \cdot b_1^{e_2} \cdot b_3^{e_2} \cdot b_4^{e_2} \cdot b_2^{e_3}$

would be computed as

$$b_1^{e_2} \cdot b_2^{e_1+e_3} \cdot b_3^{e_1+e_2} \cdot b_4^{e_2}$$

using a vector addition chain algorithm. In the cases were a single power was to be computed, the "window method" of [Bos92] was applied.

The table shows that in the general case, where verification is done more often than signing, it is advantageous to use a small P, possibly of only one element. The length of the list R is not a problem if it is generated from a seed, as suggested above. Another advantage of using a small set P is that the list R has to change less often.

# R	#P	message	sign	verify
250	1	245	910	152
50	5	245	1512	272
5	50	245	1451	2048*
1	250	245	796	7123*
500	1	495	1035	278
50	10	495	2964*	1372*
68	1	64	819	61
17	4	64	1317	162
4	17	64	1301	659*
Fig. 3. Performance for different size of R and P.				

The influence of the modulus size and prime size on the performance is shown is

Figure 4. In this table, the size of R is set to 50 elements, while the sets P contain 5 elements each. The number of multiplications for signing depends on the size of the modulus only, while the number of multiplications for verifying depends on the size of the prime numbers only. Although it saves a little time during the signing to use a shorter modulus, we suggest using a modulus of 668 bits, since the current technology already allows factoring numbers of up to 351 bits.

The size of the primes in the sets P determines the verification time. Choosing smaller primes increases the speed of verification, but allows fewer signatures before a new list R is needed.

madulus sina	Laionina	prime size	verifying
modulus size	signing	10	171
512	1172	20	272
668 1512	30	381	

Fig. 4. Performance for different sizes of modulus and primes.

If the elements of A are precomputed, signing takes #A/2-1 multiplications. The precomputation takes about 796 #A multiplications, so precomputation is only effective if there is plenty of time for doing it.

For extremely fast verification of signatures, we choose a list R of 68 elements, generated from a seed number that is part of the signature, and $P = \{3\}$. For these parameters, the message to be signed is 64 bits (8 bytes). This allows verification of a signature in only 35 modular multiplications, plus the time to generate the elements of R. Signing takes about 819 multiplications. Using precomputation, signing takes 33 multiplications, but about 55000 multiplications for the precomputation.

Proof of unforgeability

We prove that the signature scheme is "existentially unforgeable under an adaptive chosen-message attack". This means that, under the RSA assumption, if an attacker can influence the signer to sign any number of messages of his liking, he cannot forge new signatures in polynomial time, even if the messages depend on the signatures on earlier messages.

The main theorem used to prove unforgeability of the signature system is proved by Jan-Hendrik Evertse and Eugène van Heijst in [EH90], and is a generalization of a theorem by Adi Shamir [Sha83]. The theorem is about computing a product of RSA roots with a given modulus if a set of products of signatures is known. Under the RSA assumption, the theorem states that if a set of products of roots is known, the only new products of roots that can be constructed in polynomial time are those that can be computed using multiplication and division.

One assumption we make is that the attacker cannot combine the signatures of different participants, because they have different moduli. This is still an open problem. This assumption allows us to use the results of [EH90].

In our situation, we assume an attacker who knows many signature products S from a participant. These products can be written as products of roots of elements of **R**:

$$r_1^{x_1}r_2^{x_2}r_3^{x_3}\cdots r_{*R}^{x_{*R}}$$

where the numbers x_i are rational numbers. The theorem of [EH90] states that if we interpret the x as vectors, the only new products that can be computed by the attacker correspond to linear combinations of these vectors. What remains to be proved is that linear combinations of these vectors do not give products that the attacker can use for new signatures.

The denominators of the rational numbers x_i are products of primes from the set P of the corresponding signature, since the x_i are sums of the form $\frac{1}{P_1} + \frac{1}{P_2} + \cdots$, where $p_i \in \mathbf{P}$. This means that we can speak of "the set of primes in a vector", meaning both the set of primes that occur in the denominators of the elements, and the set P used for generating the signature. Every signature uses another P, and the sets P do not overlap, so the sets of primes in the vectors also do not overlap. A linear combination of vectors will contain only primes that occurred in the original vectors. From this we see that combining signatures with multiplication and division will not produce a signature with a set P that is not used before.

For a set P that has already been used, the only linear combination of vectors that contains the primes of P is a multiple of the corresponding vector, because any other linear combination of vectors contains primes not in P. This means that other signature products do not help compute a new signature product with a given set P. From the definition of the signature product, we see that a power of a product cannot be a signature on another message, so this method also yields no new signatures for the attacker.

Note that if m is a one-way hash function of a message, signatures on other messages can be forged if the hash function is broken. This is of course a separate problem from the security of the signature scheme.

From the above we conclude that an attacker cannot, under the RSA assumption, produce a signature product that is not already computed by the signer. This finishes the proof that the signature scheme is secure.

Conclusion

It was already known that a signature with provable unforgeability existed under the factoring assumption. Our scheme, based on the modular root assumption, improves on the scheme in the literature on several points: signatures are smaller, while signing and verification use much less memory and computation. The new scheme has a large degree of flexibility, allowing the signing of both long and short messages by varying the parameters.

References

- [BCDP90] J. F. Boyar, D. Chaum, I. B. Damgård and T. Pedersen: Convertible Underiable Signatures, Advances in Cryptology: Proc. Crypto '90 (Santa Barbara, CA, August 1990), to be published.
- [BD85] E. F. Brickelt and J. M. DeLaurentis: An Attack on a Signature Scheme proposed by Okamoto and Shiraishi, Advances in Cryptology: Proc. Crypto '85 (Santa Barbara, CA, August 1985), pp. 28-32.
- [Bet83] T. Beth: A fiat-Shamir-like Authentication Protocol for the ElGanal Scheme, Advances in Cryptology: Proc. Eurocrypt '88 (Daves, Switzerland, May 1988), pp. 77-86.
- [BM88] M. Bellare and S. Micali: How to Sign Given any Trapdoor Function, Advances in Cryptology: Proc. Crypto '88 (Santa Barbara, CA, August 1988), pp. 200-215.
- [Bos92] J. N. E. Bos: *Practical Privacy*, dissentation of the Eindhoven University of Technology, march 1992.
- [CA89] D. Chaum and H. van Antwerpen: Undeniable Signatures, Advances in Cryptology: Proc. Crypto '89 (Santa Barbara, CA, August 1989), pp. 212-216.
- [Cha82] D. Chaum: Blind Signatures for Untraceable Payments, Advances in Cryptology: Proc. Crypto '82 (Santa Barbara, CA, August 1982), pp. 199-203.
- [Cha90] D. Chaum: Zero-knowledge Undeniable Signatures, Advances in Cryptology: Proc. Eurocrypt '90 (Århus, Denmark, May 1990), pp. 458-464.
- [CHP91] D. Chaum, E. van Heijst, and B. Pfitzmann: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer, Advances of Cryptology: Proc. Crypto '91 (Santz Barbara, August 1991), to be published.
- [CR90] D. Chaum and S. Roljakkers: Unconditionally Secure Digital Signatures, Advances in Cryptology: Proc. Crypto '90 (Santa Barbara, CA, August 1990), pp. 209-217.
- [Den84] D. E. R. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems, Comm. ACM 27 (No. 4, April 1984), pp. 388-392.
- [DH76] W. Diffe and M. E. Hellman: New Directions in Cryptography, IEEE Trans. Information Theory IT-22 (No. 6, November 1976), pp. 644-654.
- [DLM82] R. DeMillo, N. Lynch, and M. Merritt: Cryptographic Protocols, Proc. 14th ACM Symp. Theory of Computing (San Fransisco, CA, May 1982), pp. 383-400.

[EAKMM85]

D. Estes, L. M. Adleman, K. Kompella, K. McCurley, and G. L. Miller: Breaking the Ong-Schnorr-Shamir Signature Scheme for Quadratic Number fields, Advances in Cryptology: Proc. Crypto '85 (Santa Barbara, CA, August 1985), pp. 3-13.