

# How to Leak a Secret

Ronald L. Rivest  
Laboratory for Computer Science  
Massachusetts Institute of Technology  
Cambridge, MA 02139  
rivest@mit.edu

Adi Shamir  
Applied Math Department  
Weizmann Institute of Science  
shamir@wisdom.weizmann.ac.il

Yael Tauman  
Applied Math Department  
Weizmann Institute of Science  
tauman@wisdom.weizmann.ac.il

June 4, 2001

## Abstract

We introduce the notion of a *ring signature*: a digital signature that specifies a set of possible signers, such that the verifier can't tell which member actually produced the signature. Unlike group signatures, ring signatures have no group managers, no setup procedures, and no coordination: any user can sign on behalf of any set to which he belongs, and he can choose a new set for each message without getting the consent or assistance of the other members. The only requirement is that each possible signer is already using some public key signature scheme, such as RSA. Ring signatures provide an elegant way to leak authoritative secrets in an anonymous way, to implement designated-verifier signature schemes which can authenticate emails without undesired side effects, and to solve other problems in multiparty computations. Finally, we propose a ring signature scheme which is provably secure in the random oracle model, provides unconditional anonymity, and is exceptionally efficient: adding each ring member increases the cost of signing or verifying by a single modular multiplication and a single symmetric encryption.

**Keywords:** signature scheme, ring signature scheme, signer-ambiguous signature scheme, group signature scheme, designated verifier signature scheme.

## 1 Introduction

We present an efficient signature scheme that is *signer-ambiguous*: a verifier does not learn who the actual signer is, but only that he belongs to a certain set of possible signers.

The similar notion of *group signatures* was introduced in 1991 by Chaum and van Heyst [2]. However, in our proposed scheme there are no central group managers, no prearranged groups of users, and no procedures for setting, changing, or deleting groups. We call such signatures “ring signatures” instead of “group signatures” since rings are geometric regions with uniform periphery and no center. In a ring signature scheme, any user can declare for any message any set of possible signers which includes himself, and produce the signature entirely by himself using only his secret key and the others' public keys. In particular, the other possible signers may be completely unaware that their public keys are used by a stranger to produce such signatures.

The main purpose of a ring signature is to provide (limited) anonymity for the signer, by making it impossible to determine who among the possible signers is the actual one. All the previously published anonymization techniques were computationally inefficient for large sets, since both the signature generation and the signature verification required a number of modular exponentiations which grew linearly with the number of possible signers. In this paper we introduce a new construction paradigm which adds only one modular multiplication and one symmetric encryption for each possible signer, and produces a ring signature scheme which is unconditionally signer-ambiguous and provably secure in the random oracle model.

## 2 Definitions and Applications

### 2.1 Ring signatures

**Terminology:** We call a set of *possible signers* a *ring*. We call the ring member who produces the actual signature the *signer* and each of the other ring members a *non-signer*.

We assume that each possible signer is associated (via a PKI directory or certificate) with a public key  $P_k$  that defines his signature scheme and specifies his verification key. The corresponding secret key (which is used to generate regular signatures) is denoted by  $S_k$ . The general notion of a ring signature scheme does not require any special properties of these individual signing schemes, but our simplest construction assumes that they use trapdoor one-way permutations (such as the RSA functions) to generate and verify signatures.

A ring signature scheme is defined by two procedures:

- **ring-sign**( $m, P_1, P_2, \dots, P_r, s, S_s$ ) which produces a ring signature  $\sigma$  for the message  $m$ , given the public keys  $P_1, P_2, \dots, P_r$  of the  $r$  ring members, together with the secret key  $S_s$  of the  $s$ -th member (who is the actual signer).
- **ring-verify**( $m, \sigma$ ) which accepts a message  $m$  and a signature  $\sigma$  (which includes the public keys of all the possible signers), and outputs either TRUE OR FALSE.

A ring signature scheme is *set-up free*: The signer does not need the knowledge, consent, or assistance of the other ring members to put them in the ring - all he needs is knowledge of their regular public keys. Different members can use different independent public key signature schemes, with different key and signature sizes. Verification must satisfy the usual soundness and completeness conditions, but in addition we want the signatures to be *signer-ambiguous* in the sense that the verifier should be unable to determine the identity of the actual signer in a ring of size  $r$  with probability greater than  $1/r$ . This limited anonymity can be either *computational* or *unconditional*. Our main construction provides unconditional anonymity in the sense that even an infinitely powerful adversary with access to an unbounded number of chosen-message signatures produced by the same ring member cannot guess his identity with any advantage, and cannot link additional signatures to the same signer.

### 2.2 Leaking secrets

To motivate the title for this paper, suppose that Bob (also known as “Deep Throat”) is a member of the cabinet of Lower Kryptonite, and that Bob wishes to leak a juicy fact to a journalist about the escapades of the Prime Minister, in such a way that Bob remains anonymous, yet such that the journalist is convinced that the leak was indeed from a cabinet member.

Bob cannot send to the journalist a standard digitally signed message, since such a message, although it convinces the journalist that it came from a cabinet member, does so by directly revealing Bob's identity.

It also doesn't work for Bob to send the journalist a message through a standard anonymizer, since the anonymizer strips off all source identification and authentication: the journalist would have no reason to believe that the message really came from a cabinet member at all.

A standard group signature scheme does not solve the problem, since it requires the prior cooperation of the other group members to set up, and leaves Bob vulnerable to later identification by the group manager, who may be controlled by the Prime Minister.

The correct approach is for Bob to send the story to the journalist through an anonymizer, signed with a ring signature scheme that names each cabinet member (including himself) as a ring member. The journalist can verify the ring signature on the message, and learn that it definitely came from a cabinet member. He can even post the ring signature in his paper or web page, to prove to his readers that the juicy story came from a reputable source. However, neither he nor his readers can determine the actual source of the leak, and thus the whistleblower has perfect protection even if the journalist is later forced by a judge to reveal his source.

### 2.3 Designated verifier signature schemes

A designated verifier signature scheme is a signature scheme in which signatures can only be verified by a single "designated verifier" chosen by the signer. This concept was first introduced by Jakobsson Sako and Impagliazzo at Eurocrypt 96 [5]. A typical application is to enable users to authenticate casual emails without being legally bound to their contents. For example, two companies may exchange drafts of proposed contracts. They wish to add to each email an authenticator, but not a real signature which can be shown to a third party (immediately or years later) as proof that a particular draft was proposed by the other company. A designated verifier scheme can thus be viewed as a "light signature scheme" which can authenticate messages to their intended recipients without having the nonrepudiation property.

One approach would be to use zero knowledge interactive proofs, which can only convince their verifiers. However, this requires interaction and is difficult to integrate with standard email systems and anonymizers. We can use non-interactive zero knowledge proofs, but then the authenticators become signatures which can be shown to third parties. Another approach is to agree on a shared secret symmetric key  $k$ , and to authenticate each contract draft by appending a message authentication code (MAC) for the draft computed with key  $k$ . A third party would have to be shown the secret key to validate a MAC, and even then he wouldn't know which of the two companies computed the MAC. However, this requires an initial set-up procedure, in which we still face the problem of authenticating the emailed choice of  $k$  without actually signing it.

A designated verifier scheme provides a simple solution to this problem: company A can sign each draft it sends, naming company B as the designated verifier. This can be easily achieved by using a ring signature scheme with companies A and B as the ring members. Just as with a MAC, company B knows that the message came from company A (since no third party could have produced this ring signature), but company B cannot prove to anyone else that the draft of the contract was signed by company A, since company B could have produced this draft by itself. Unlike the case of MAC's, this scheme uses public key cryptography, and thus A can send unsolicited email to B signed with the ring signature without any preparations, interactions, or secret key exchanges. By using our proposed ring signature scheme, we can turn standard signature schemes into designated verifier schemes which can be added at almost no cost as an extra option to any email system.

## 2.4 Efficiency of our ring signature scheme

When based on Rabin or RSA signatures, our ring signature scheme is particularly efficient:

- signing requires one modular exponentiation, plus one or two modular multiplications for each non-signer.
- verification requires one or two modular multiplications for each ring member.

In essence, generating or verifying a ring signature costs the same as generating or verifying a regular signature plus an extra multiplication or two for each non-signer, and thus the scheme is truly practical even when the ring contains hundreds of members. It is two to three orders of magnitude faster than Camenisch's scheme, whose claimed efficiency is based on the fact that it is 4 times faster than earlier known schemes (see bottom of page 476 in his paper [1]). In addition, a Camenisch-like scheme uses linear algebra in the exponents, and thus requires all the members to use the same prime modulus  $p$  in their individual signature schemes. One of our design criteria is that the signer should be able to assemble an arbitrary ring without any coordination with the other ring members. In reality, if one wants to use other users' public keys, they are much more likely to be RSA keys, and even if they are based on discrete logs, different users are likely to have different moduli  $p$ . The only realistic way to arrange a Camenisch-like signature scheme is thus to have a group of consenting parties.

Note that the size of any ring signature must grow linearly with the size of the ring, since it must list the ring members; this is an inherent disadvantage of ring signatures as compared to group signatures that use predefined groups.

## 3 The Proposed Ring Signature Scheme (RSA version)

To simplify the presentation and proof, we first describe a ring signature scheme in which all the ring members use RSA [7] as their individual signature schemes. The same construction can be used for any other trapdoor one way permutation, but we have to modify it slightly in order to use trapdoor one way functions (as in, for example, Rabin's signature scheme [6]).

Suppose that Alice wishes to sign a message  $m$  with a ring signature for the ring of  $r$  individuals  $A_1, A_2, \dots, A_r$ , where the signer Alice is  $A_s$ , for some value of  $s$ ,  $1 \leq s \leq r$ .

### 3.1 RSA trap-door permutations

Each ring member  $A_i$  has an RSA public key  $P_i = (n_i, e_i)$  which specifies the trapdoor one-way permutation  $f_i$  of  $\mathbf{Z}_{n_i}$ :

$$f_i(x) = x^{e_i} \pmod{n_i} .$$

We assume that only  $A_i$  knows how to compute the inverse permutation  $f_i^{-1}$  efficiently, using trap-door information; this is the original Diffie-Hellman model [3] for public-key cryptography.

#### Extending trap-door permutations to a common domain

The trap-door RSA permutations of the various ring members will have domains of different sizes (even if all the moduli  $n_i$  have the same number of bits). This makes it awkward to combine the individual signatures, and thus we extend all the trap-door permutations to have as their common domain the same set  $\{0, 1\}^b$ , where  $2^b$  is some power of two which is larger than all the moduli  $n_i$ 's.

For each trap-door permutation  $f_i$  over  $\mathbf{Z}_{n_i}$ , we define the extended trap-door permutation  $g_i$  over  $\{0, 1\}^b$  in the following way. For any  $b$ -bit input  $m$  define nonnegative integers  $q_i$  and  $r_i$  so that  $m = q_i n_i + r_i$  and  $0 \leq r_i < n_i$ . Then

$$g_i(m) = \begin{cases} q_i n_i + f_i(r_i) & \text{if } (q_i + 1)n_i \leq 2^b \\ m & \text{else.} \end{cases}$$

Intuitively,  $g_i$  is defined by using  $f_i$  to operate on the low-order digit of the  $n_i$ -ary representation of  $m$ , leaving the higher order digits unchanged. The exception is when this might cause a result larger than  $2^b - 1$ , in which case  $m$  is unchanged. If we choose a sufficiently large  $b$  (e.g. 160 bits larger than any of the  $n_i$ ), the chance that a randomly chosen  $m$  is unchanged by the extended  $g_i$  becomes negligible. (A stonger but more expensive approach, which we don't need, would use instead of  $g_i(m)$  the function  $g'_i(m) = g_i((2^b - 1) - g_i(m))$  which can modify all its inputs). The function  $g_i$  is clearly a permutation over  $\{0, 1\}^b$ , and it is a one-way trap-door permutation since only someone who knows how to invert  $f_i$  can invert  $g_i$  efficiently on more than a negligible fraction of the possible inputs.

### 3.2 Symmetric encryption

We assume the existence of a publicly defined symmetric encryption algorithm  $E$  such that for any key  $k$  of length  $l$ , the function  $E_k$  is a permutation over  $b$ -bit strings. Here we use the random (permutation) oracle model which assumes that all the parties have access to an oracle that provides truly random answers to new queries of the form  $E_k(x)$  and  $E_k^{-1}(y)$ , provided only that they are consistent with previous answers and with the requirement that  $E_k$  be a permutation.

### 3.3 Hash functions

We assume the existence of a publicly defined collision-resistant hash function  $h$  that maps arbitrary inputs to strings of length  $l$ , which are used as keys for  $E$ . We model  $h$  as a random oracle. (Since  $h$  need not be a permutation, different queries may have the same answer, and we will disallow “ $h^{-1}$ ” queries.)

### 3.4 Combining functions

We define a family of keyed “combining functions”  $C_{k,v}(y_1, y_2, \dots, y_r)$  which take as input a key  $k$ , an initialization value  $v$ , and arbitrary values  $y_1, y_2, \dots, y_r$  in  $\{0, 1\}^b$ . Each such combining function uses  $E_k$  as a sub-procedure, and produces as output a value  $z$  in  $\{0, 1\}^b$  such that given any fixed values for  $k$  and  $v$ , we have the following properties.

1. **Permutation on each input:** For each  $s$ ,  $1 \leq s \leq r$ , and for any fixed values of all the other inputs  $y_i$ ,  $i \neq s$ , the function  $C_{k,v}$  is a one-to-one mapping from  $y_s$  to the output  $z$ .
2. **Efficiently solvable for any single input:** For each  $s$ ,  $1 \leq s \leq r$ , given a  $b$ -bit value  $z$  and values for all inputs  $y_i$  except  $y_s$ , it is possible to efficiently find a  $b$ -bit value for  $y_s$  such that  $C_{k,v}(y_1, y_2, \dots, y_r) = z$ .
3. **Infeasible to solve verification equation for all inputs without trap-doors:** Given  $k, v$ , and  $z$ , it is infeasible for an adversary to solve the equation

$$C_{k,v}(g_1(x_1), g_2(x_2), \dots, g_r(x_r)) = z \tag{1}$$

for  $x_1, x_2, \dots, x_r$ , (given access to each  $g_i$ , and to  $E_k$ ) if the adversary can't invert any of the trap-door functions  $g_1, g_2, \dots, g_r$ .

For example, the function

$$C_{k,v}(y_1, y_2, \dots, y_r) = y_1 \oplus y_2 \oplus \dots \oplus y_r$$

(where  $\oplus$  is the exclusive-or operation on  $b$ -bit words) satisfies the first two of the above conditions, and can be kept in mind as a candidate combining function. Indeed, it was the first one we tried. But it fails the third condition since for any choice of trapdoor one-way permutations  $g_i$ , it is possible to use linear algebra when  $r$  is large enough to find a solution for  $x_1, x_2, \dots, x_r$  without inverting any of the  $g_i$ 's. The basic idea of the attack is to choose a random value for each  $x_i$ , and to compute each  $y_i = g_i(x_i)$  in the easy forward direction. If the number of values  $r$  exceeds the number of bits  $b$ , we can find with high probability a subset of the  $y_i$  bit strings whose XOR is any desired  $b$ -bit target  $z$ . However, our goal is to represent  $z$  as the XOR of all the values  $y_1, y_2, \dots, y_r$  rather than as a XOR of a random subset of these values. To overcome this problem, we choose for each  $i$  *two* random values  $x'_i$  and  $x''_i$ , and compute their corresponding  $y'_i = g_i(x'_i)$  and  $y''_i = g_i(x''_i)$ . We then define for each  $i$   $y'''_i = y'_i \oplus y''_i$ , and modify the target value to  $z' = z \oplus y'_1 \oplus y'_2, \dots \oplus y'_r$ . We use the previous algorithm to represent  $z'$  as a XOR of a random subset of  $y'''_i$  values. After simplification, we get a representation of the original  $z$  as the XOR of a set of  $r$  values, with exactly one value chosen from each pair  $(y'_i, y''_i)$ . By choosing the corresponding value of either  $x'_i$  or  $x''_i$ , we can solve the verification equation without inverting any of the trapdoor one-way permutations  $g_i$ . (One approach to countering this attack, which we don't explore further here, but may in our full paper, is to let  $b$  grow with  $r$ .)

Even worse problems can be shown to exist in other natural combining functions such as addition mod  $2^b$ . Assume that we use the RSA trapdoor functions  $g_i(x_i) = x_i^3 \pmod{n_i}$  where all the moduli  $n_i$  have the same size  $b$ . It is known [4] that any nonnegative integer  $z$  can be efficiently represented as the sum of exactly nine nonnegative integer cubes  $x_1^3 + x_2^3 + \dots + x_9^3$ . If  $z$  is a  $b$ -bit target value, we can expect each one of the  $x_i^3$  to be slightly shorter than  $z$ , and thus their values are not likely to be affected by reducing each  $x_i^3$  modulo the corresponding  $b$ -bit  $n_i$ . Consequently, we can solve the verification equation  $(x_1^3 \pmod{n_1}) + (x_2^3 \pmod{n_2}) \dots + (x_9^3 \pmod{n_9}) = z \pmod{2^b}$  with nine RSA permutations without inverting any one of them.

Our proposed combining function utilizes the symmetric encryption function  $E_k$  as follows:

$$C_{k,v}(y_1, y_2, \dots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\dots \oplus E_k(y_1 \oplus v) \dots)))) .$$

This function is applied to the sequence  $(y_1, y_2, \dots, y_r)$ , where  $y_i = g_i(x_i)$ , as shown in Figure 1; the resulting function is provably secure in the random oracle model.

It is clearly a permutation on each input, since the XOR,  $g_i$ , and  $E_k$  functions are permutations. In addition, it is efficiently solvable for any single input since knowledge of  $k$  makes it possible to run the evaluation forwards from the initial  $v$  and backwards from the final  $z$  in order to uniquely compute any missing value  $y_i$ . This function can be used to verify signatures by using a hashed version of  $m$  to choose the symmetric key  $k$ , and forcing the output  $z$  to be equal to the input  $v$ . This consistency condition  $C_{k,v}(y_1, y_2, \dots, y_r) = v$  bends the line into the ring shape shown in Fig. 2.

A slightly more compact ring signature variant can be obtained by always selecting 0 as the "glue value"  $v$ . This variant is also secure, but we prefer the total ring symmetry of our main proposal.

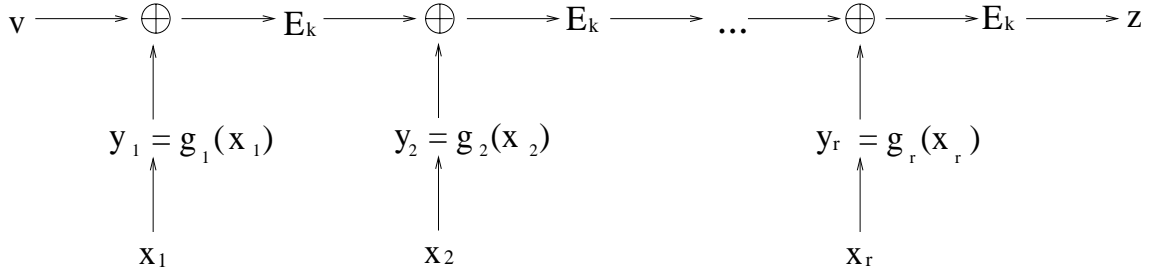


Figure 1: An illustration of the proposed combining function

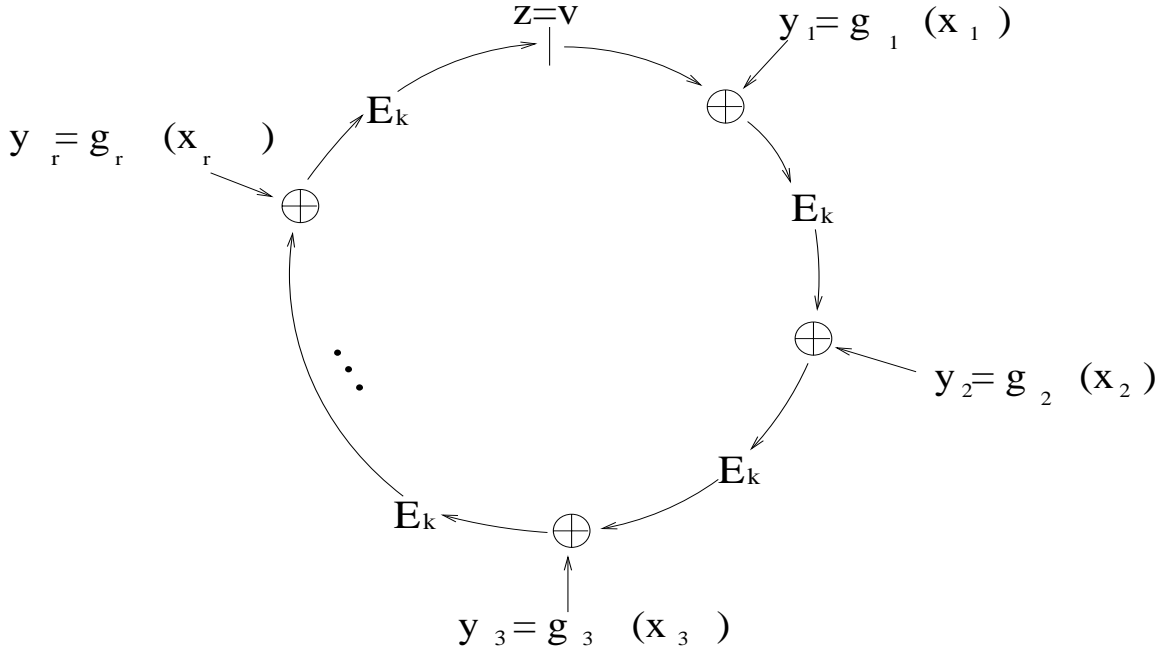


Figure 2: Ring signatures

We now formally describe the signature generation and verification procedures:

**Generating a ring signature:**

Given the message  $m$  to be signed, his secret key  $S_s$ , and the sequence of public keys  $P_1, P_2, \dots, P_r$  of all the ring members, the signer computes a ring signature as follows.

1. **Choose a key:** The signer first computes the symmetric key  $k$  as the hash of the message  $m$  to be signed:

$$k = h(m)$$

(We have also looked at the interesting variant  $k = h(m, P_1, \dots, P_r)$ ; the simpler construction works fine, however.)

2. **Pick a random glue value:** Second, the signer picks an initialization (or “glue”) value  $v$  uniformly at random from  $\{0, 1\}^b$ .
3. **Pick random  $x_i$ 's:** Third, the signer picks random  $x_i$  for all the other ring members  $1 \leq i \leq r$ ,

$i \neq s$  uniformly and independently from  $\{0, 1\}^b$ , and computes

$$y_i = g_i(x_i) .$$

**4. Solve for  $y_s$ :** Fourth, the signer solves the following ring equation for  $y_s$ :

$$C_{k,v}(y_1, y_2, \dots, y_r) = v .$$

By assumption, given arbitrary values for the other inputs, there is a unique value for  $y_s$  satisfying the equation, which can be computed efficiently.

**5. Invert the signer's trap-door permutation:** Fifth, the signer uses his knowledge of his trapdoor in order to invert  $g_s$  on  $y_s$  to obtain  $x_s$ :

$$x_s = g_s^{-1}(y_s) .$$

**6. Output the ring signature:** The signature on the message  $m$  is defined to be the  $(2r + 1)$ -tuple:

$$(P_1, P_2, \dots, P_r; v; x_1, x_2, \dots, x_r) .$$

**Verifying a ring signature:**

A verifier can verify an alleged signature

$$(P_1, P_2, \dots, P_r; v; x_1, x_2, \dots, x_r) .$$

on the message  $m$  as follows.

**1. Apply the trap-door permutations:** First, for  $i = 1, 2, \dots, r$  the verifier computes

$$y_i = g_i(x_i) .$$

**2. Obtain  $k$ :** Second, the verifier hashes the message to compute the encryption key  $k$ :

$$k = h(m) .$$

**3. Verify the ring equation:** Finally, the verifier checks that the  $y_i$ 's satisfy the fundamental equation:

$$C_{k,v}(y_1, y_2, \dots, y_r) = v . \tag{2}$$

If the ring equation (2) is satisfied, the verifier accepts the signature as valid. Otherwise the verifier rejects.

### 3.5 Security

The identity of the signer is unconditionally protected with our ring signature scheme. To see this, note that for each  $k$  and  $v$  the ring equation has exactly  $(2^b)^{(r-1)}$  solutions, and all of them can be chosen by the signature generation procedure with equal probability, regardless of the signer's identity. This argument does not depend on any complexity-theoretic assumptions or on the randomness of the oracle.

The soundness of the ring signature scheme must be computational, since ring signatures cannot be stronger than the individual signature scheme used by the possible signers. Our goal now is to show that in the random oracle model, any forging algorithm  $A$  which can generate with non-negligible probability a new ring signature for  $m$  by analysing polynomially many ring signatures for other chosen messages  $m_j \neq m$ , can be turned into an algorithm  $B$  which inverts one of the trapdoor one-way functions  $g_i$  on random inputs  $y$  with non-negligible probability.

Algorithm  $A$  accepts the public keys  $P_1, P_2, \dots, P_r$  (but not any of the corresponding secret keys) and is given oracle access to  $h, E, E^{-1}$ , and to a ring signing oracle. It can work adaptively, querying the oracles at arguments which may depend on previous answers. Eventually, it must produce a valid ring signature on a new message which was not presented to the signing oracle, with a non-negligible probability (over the random answers of the oracles and its own random tape).

Algorithm  $B$  uses algorithm  $A$  as a black box, but has full control over its oracles.  $A$  must query the oracle about all the symmetric encryptions along the forged ring signature of  $m$  (otherwise the probability of satisfying the ring equation becomes negligible). Without loss of generality, we can assume that each one of these  $r$  symmetric encryptions is queried either in the "clockwise"  $E_k$  direction or in the "counterclockwise"  $E_k^{-1}$  direction, but not in both directions since this is redundant. When  $A$  makes its polynomially many queries of  $E_k$  and  $E_k^{-1}$  with various keys  $k = h(m)$ ,  $B$  can guess which  $k$  will be involved in the actual forgery with non-negligible probability, but it cannot guess which subset of  $r$  queries will be used in the final forgery and in which order they will occur along the satisfied ring equation since there are too many possibilities.

Algorithm  $B$  can easily simulate the ring signing oracle for all the other  $m_j$  by providing random vectors  $(v, x_1, x_2, \dots, x_r)$  as their ring signatures, and adjusting the random answers for queries of the form  $E_{h(m_j)}$  and  $E_{h(m_j)}^{-1}$  to support the correctness of the ring equation for these messages. Note that  $A$  cannot ask relevant oracle questions which will limit  $B$ 's freedom of choice before providing  $m_j$  to the signing oracle since all the values along the actual ring signature (including  $v$ ) are chosen randomly by  $B$  when it provides the requested signature, and cannot be guessed in advance by  $A$ . In addition, we use the assumption that  $h$  is collision resistant to show that  $E$  and  $E^{-1}$  queries with key  $k_j = h(m_j)$  will not constrain the answers to  $E$  and  $E^{-1}$  queries with key  $k = h(m)$  which will be used in the final forgery, since they use different keys.

The goal of algorithm  $B$  is to compute for some  $i$   $x_i = g_i^{-1}(y)$  for random inputs  $y$ 's with non-negligible probability. This will reduce the security of the ring signature to the security of the individual signature schemes. The basic idea of the reduction is to slip this random  $y$  as the "gap" between the output and input values of two cyclically consecutive  $E$ 's along the ring equation of the final forgery, which forces  $A$  to close the gap by providing the corresponding  $x_i$  in the generated signature. Note that  $y$  is a random value which is known to  $B$  but not to  $A$ , and thus  $A$  cannot "recognize the trap" and refuse to sign the corresponding messages.

The main difficulty is that  $A$  can close gaps between  $E$  values not only by inverting trapdoor one-way functions, but also by evaluating these functions in the easy forward direction (as done by the real signer in the generation of ring signatures). To overcome this difficulty, we note that in any valid ring signature produced by  $A$ , there must be a gap somewhere between two cyclically

consecutive occurrences of  $E$  in which the queries were computed in one of the following three ways:

- The oracle for the  $i$ -th  $E$  was queried in the “clockwise” direction and the oracle for the  $i + 1$ -st  $E$  was queried in the “counterclockwise” direction.
- Both  $E$ ’s were queried in the “clockwise” direction, but the  $i$ -th  $E$  was queried after the  $i + 1$ -st  $E$ .
- Both  $E$ ’s were queried in the “counterclockwise” direction, but the  $i$ -th  $E$  was queried before the  $i + 1$ -st  $E$ .

In all these cases,  $B$  can provide a random answer to the later query which is based on his knowledge of input and output of the earlier query in such a way that the XOR of the values across the gap is the desired  $y$ . This will force  $A$  to compute the corresponding  $g_i^{-1}(y)$  in order to fill in this gap in its final ring signature.

$B$  does not know which queries will be these cyclically consecutive queries in the forged ring signature, and thus he has to guess their identity. However, he has to make only two guesses and thus the probability of guessing correctly is  $1/Q^2$  where  $Q$  is the total number of queries made by the forger  $A$ . Consequently,  $B$  will manage to compute  $g_i^{-1}(y)$  for a random  $y$  with non-negligible probability.

When the trapdoor one-way functions  $g_i$  are RSA functions, we can slightly strengthen the result. Since RSA is homomorphic, we can randomize  $y$  by computing  $y' = y * t^{e_i} \pmod{n_i}$  for a randomly chosen  $t$ . By using  $y'$  instead of  $y$ , we can show that successful forgeries of ring signatures can be used to extract modular roots from particular numbers such as  $y = 2$ , and not just from random inputs  $y$ . This is not necessarily true for other trapdoor functions, since the forger  $A$  can intentionally decide not to produce any forgeries in which one of the gaps between cyclically consecutive  $E$  functions happens to be 2.

## 4 Our Ring Signature Scheme (Rabin version)

Rabin’s public-key cryptosystem [6] has more efficient signature verification than RSA, since verification involves squaring rather than cubing, which reduces the number of modular multiplications from 2 to 1. However, we need to deal with the fact that the Rabin mapping  $f_i(x_i) = x_i^2 \pmod{n_i}$  is not a permutation over  $\mathbf{Z}_{n_i}^*$ , and thus only one quarter of the messages can be signed, and those which can be signed have multiple signatures.

The operational fix is the natural one: when signing, change your last random choice of  $x_{s-1}$  if  $g_s^{-1}(y_s)$  is undefined. Since only one trapdoor one-way function has to be inverted, the signer should expect on average to try four times before succeeding in producing a ring signature. The complexity of this search is essentially the same as in the case of regular Rabin signatures, regardless of the size of the ring.

A more important difference is in the proof of unconditional anonymity, which relied on the fact that all the mappings were permutations. When the  $g_i$  are not permutations, there can be noticeable differences between the distribution of randomly chosen and computed  $x_i$  values in given ring signatures. This could lead to the identification of the real signer among all the possible signers, and can be demonstrated to be a real problem in many concrete types of trapdoor one-way functions.

We overcome this difficulty in the case of Rabin signatures with the following simple observation:

**Theorem 1** *Let  $S$  be a given finite set of “marbles” and let  $B_1, B_2, \dots, B_n$  be disjoint subsets of  $S$  (called “buckets”) such that all non-empty buckets have the same number of marbles, and every marble in  $S$  is in exactly one bucket. Consider the following sampling procedure: pick a bucket at random until you find a non-empty bucket, and then pick a marble at random from that bucket. Then this procedure picks marbles from  $S$  with uniform probability distribution.*

**Proof:** Trivial. ■

Rabin’s functions  $f_i(x_i) = x_i^2 \pmod{n_i}$  are extended to functions  $g_i(x_i)$  over  $\{0, 1\}^b$  in the usual way. Both the marbles and the buckets are all the  $b$ -bit numbers  $u = q_i n_i + r_i$  in which  $r_i \in \mathbf{Z}_{n_i}^*$  and  $(q_i + 1)n_i \leq 2^b$ . Each marble is placed in the bucket to which it is mapped by the extended Rabin mapping  $g_i$ . We know that each bucket contains either zero or four marbles, and the lemma implies that the sampled distribution of the marbles  $x_i$  is exactly the same regardless of whether they were chosen at random or picked at random among the computed inverses in a randomly chosen bucket. Consequently, even an infinitely powerful adversary cannot distinguish between signers and nonsigners by analysing actual ring signatures produced by one of the possible signers.

## 5 Discussions and Generalizations

The new notion<sup>1</sup> of ring signatures has many interesting extensions and modifications. We discuss some of them below.

Ring signatures with  $r = 1$  can be viewed as a randomized version of Rabin’s signature scheme. As shown in Fig. 3, the verification condition can be written as  $(x^2 \pmod{n}) = v \oplus E_{h(m)}^{-1}(v)$ . The right hand side is essentially a hash of the message  $m$ , randomized by the choice of  $v$ .

Ring signatures with  $r = 2$  have the ring equation:

$$E_{h(m)}(x_2^2 \oplus E_{h(m)}(x_1^2 \oplus v)) = v$$

(see Fig. 3). A simpler ring equation (which is not equivalent but has the same security properties) is:

$$(x_1^2 \pmod{n_1}) = E_{h(m)}(x_2^2 \pmod{n_2})$$

where the modular squares are extended to  $\{0, 1\}^b$  in the usual way. This is the recommended method for implementing designated verifier signatures in email systems, where  $n_1$  is the public key of the sender and  $n_2$  is the public key of the recipient.

In regular ring signatures it is provably impossible for an adversary to expose the signer’s identity. However, there may be cases in which the signer himself wants to have the option of later proving his authorship of the anonymized email (e.g., if he is successful in toppling the disgraced Prime Minister). Yet another possibility is that the signer A wants to initially use  $\{A, B, C\}$  as the list of possible signers, but later prove that C is *not* the real signer. There is a simple way to implement these options, by choosing the  $x_i$  values for the nonsigners in a pseudorandom rather than truly random way. To show that C is *not* the author, A publishes the seed which pseudorandomly generated the part of the signature associated with C. To prove that A *is* the signer, A can reveal

---

<sup>1</sup>While the notion of a ring signature is new, previous authors have proposed schemes that qualify as ring signature schemes. In particular, Chaum et al. [2]’s schemes three and four, and the two signature schemes in Definitions 2 and 3 of Camenisch’s paper [1], qualify as ring signature schemes. However the former schemes require zero-knowledge proofs with each signature, and the latter schemes require as many modular exponentiations as there are members in the ring.

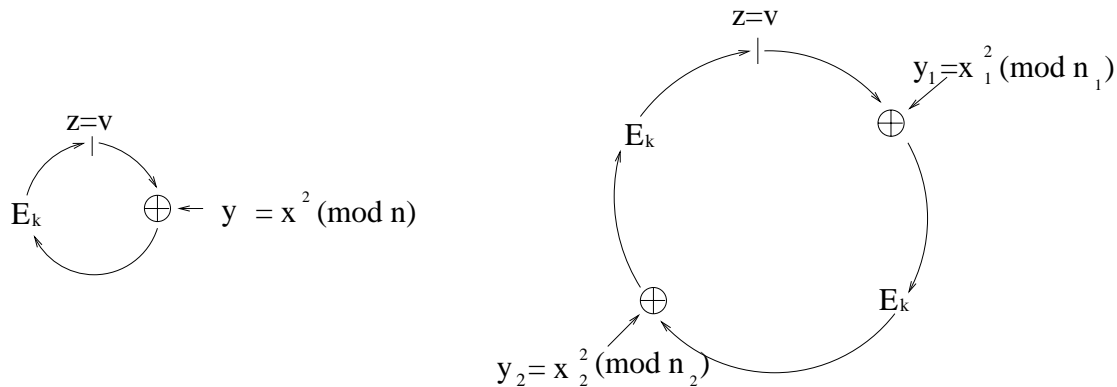


Figure 3: Rabin-based Ring Signatures with  $r = 1, 2$

a single seed which was used to generate all the nonsigners' parts of the signature. The signer A cannot misuse this technique to prove that he is not the signer since his part is computed rather than generated, and is extremely unlikely to have a corresponding seed. Note that these modified versions can guarantee only computational anonymity, since a powerful adversary can search for such proofs of nonauthorship and use them to expose the signer.

Another interesting extension is to combine ring signatures with threshold schemes (see [8]). Suppose that A and B agree to cooperate, and want to sign on behalf of the ring  $\{A, B, C\}$  in such a way that recipients will know that at least two of the three possible signers agreed to sign the message, but without revealing their identities. In the full version of this paper we describe a modification of the ring verification equation which makes it possible to achieve such extensions.

## References

- [1] Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *Advances in Cryptology - Eurocrypt '97*, pages 465–479, Berlin, 1997. Springer. Lecture Notes in Computer Science 1233.
- [2] David Chaum and Eugène Van Heyst. Group signatures. In D.W. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 257–265, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 547.
- [3] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.
- [4] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford, fifth edition, 1979.
- [5] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In Ueli Maurer, editor, *Advances in Cryptology - EuroCrypt '96*, pages 143–154, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science Volume 1070.
- [6] M. Rabin. Digitalized signatures as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, January 1979.

- [7] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [8] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.