Threshold Ring Signatures and Applications to Ad-hoc Groups.

Emmanuel Bresson[1] and Jacques Stern[1] and Michael Szydlo[2]

[1] Dépt d'informatique, École normale supérieure, 75230 Paris Cedex 05, France
{Emmanuel.Bresson,Jacques.Stern}@ens.fr
[2] RSA Laboratories, 20 Crosby Drive, Bedford, MA 01730, USA
mszydlo@rsasecurity.com

Abstract. In this paper, we investigate the recent paradigm for group signatures proposed by Rivest *et al.* at Asiacrypt '01. We first improve on their ring signature paradigm by showing that it holds under a strictly weaker assumption, namely the random oracle model rather than the ideal cipher. Then we provide extensions to make ring signatures suitable in practical situations, such as threshold schemes or ad-hoc groups. Finally we propose an efficient scheme for threshold scenarios based on a combinatorial method and provably secure in the random oracle model.

1 Introduction

In many multi-user cryptographic applications, anonymity is required to ensure that information about the user is not revealed. Typical examples are electronic voting [?,?], digital lotteries [?,?], or e-cash applications [?,?]. In these applications releasing private information is highly undesirable and may result in a large financial loss. The concept of group signatures introduced in 1991 [?], allows a registered member of a predefined group to produce anonymous signatures on behalf of the group. However, this anonymity can be revoked by an authority if needed. The extra trapdoor information stored by the authority is used to reveal the identity of the actual signer. This mechanism provides some level of security for non-signing members in case of dispute. Hence, group signatures are only the appropriate tool when members have agreed to cooperate. The distinct but related concept of *ring signature* has recently been formalized by Rivest *et al* [?]. This concept is of particular interest when the members do not agree to cooperate since the scheme requires neither a group manager, nor a setup procedure, nor the action of a non-signing member.

A ring signature specifies a set of possible signers and a proof that is intended to convince any verifier that the author of the signature belongs to this set, while hiding his identity. More precisely, these schemes differ in several ways from classical group signature schemes. First of all, for any message, a signer may add his name to any set of other users he chooses, and produce a ring signature on it which reveals only that the anonymous author (in fact, himself) belongs to this set. This is infeasible with standard group signatures where the possible signers, by definition, are registered members of the group. In particular, the non-signing members may even be completely unaware that they are involved in such signature. Secondly, the absence of a revocation manager allows *unconditional* anonymity. This is not achievable in typical group signatures for which there *must* be some trap-door information to be used by the authority. Thirdly, ring signatures can be made extremely efficient. While previously proposed group signature schemes heavily use asymmetric computations, such as zero-knowledge proofs, the ring signature scheme proposed in [?] is very fast, and thus very practical to implement.

Using ring signatures in ad-hoc groups. The steadily growing importance of portable devices and mobile applications has spawned new types of groups of interacting parties: ad-hoc groups [?,?]. The highly dynamic nature of such groups raises new challenges for networking [?]. Ad-hoc networks may be described as networks with minimal infrastructure, lacking fixed routers or stable links. Such ad-hoc networks inherently deal with spontaneous ad-hoc groups; other instances of ad-hoc groups are not dependent on the particular network infrastructure. For example, a group of users who spontaneously decide they wish to communicate sensitive data need a suite of protocols which do not involve any trusted third party or certification of any new public keys. Security goals have to be considered in this new context. Ring signatures are perfectly suited to such a setting, since no setup protocol is required to use them.

Now assume that in order to create a certain signature at least k out of the n parties need to combine their knowledge. Threshold cryptography allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature). We consider the use of ring signatures in this ad-hoc threshold setting, and begin by defining the security notions useful for protocols involving such *ad-hoc groups*. The paradigm of ring signatures provides us with first building block. It gives a recipe for a single member of an ad-hoc group to sign anonymously on behalf of the group. Our flexible construction extends the solution of [?] to solve the threshold problem; in fact it is a solution for all custom signatures on an ad-hoc group.

Contributions. In this paper we significantly improve the ring signature scheme proposed by Rivest *et al.* [?]. We first show that security can be based on strictly weaker complexity assumptions, without sacrificing efficiency. Furthermore, this greatly simplifies the security proof provided in [?].

Next, we show how the ring signature paradigm and the stand alone protocol of [?] extend to a generic building block to make new schemes, in a multi-signer threshold setting. To achieve this aim, we formalize the notion of *ad-hoc group* signature, which is also of independent interest for general multi-party protocols relying on limited public key infrastructure. We define a formal security model to deal with such extended signatures, and we provide an interesting composition theorem which may be used to prove security for classical threshold structures. In case of threshold subgroups, the result remains very efficient for small threshold values and is provably secure, based on RSA, under the strictly weaker assumption of random oracles. In that construction, we use a combinatorial notion we call *fair partition*, which is of independent interest.

Related work. Informal notions of ring signatures were discussed simultaneously with the appearance of group signatures [**?**,**?**] but the concept itself has only been formalized recently in [**?**].

Many schemes have been proposed for group signatures, offering various additional properties [**?**,**?**,**?**] as well as increasing efficiency [**?**,**?**,**?**]. The related *Witness hiding* zero knowledge proofs were treated in [**?**], and an application to group signatures (without a manager) was discussed at the end of this same article. This, and another construction [**?**] can also be seen as ring signature schemes. However, the scheme by Rivest *et al.* [**?**] is the most efficient one.

While it is well-known that theoretical very general witness-hiding signature constructions are realizable, they are also widely believed to be completely impractical. In fact, presenting an efficient general signature construction complements the general theory which, for example, tells us that arbitrary statements in $\mathcal{NP}$ can be proven in zero knowledge. Our work combines the known general techniques with some novel constructions and specializes to the case where custom signatures are required in an environment with only a pre-existing PKI. The constructions herein are adaptable to both ad-hoc and traditional threshold structures via a standard procedure which produces custom signatures, complete with a specialization of the security proof and efficiency analysis.

Thus, this work is at the crossroad of three corresponding trends in cryptography research: provable security, custom protocol design, efficiency analysis.

The organization of the rest of the paper is as follows: we review the concept of ring signature in section **??** and explain how improve it in section **??**. Next in section **??**, we extend the notion, propose a formal security model and finally prove the security of our new scheme in section **??**.

2 Overview of Ring Signatures

In this section, we follow the formalization proposed by Rivest, Shamir and Tauman in [**?**].

2.1 Definitions

One assumes that each user has received (via a PKI or a certificate) a public key P_k, for which the corresponding secret key is denoted S_k. A ring signature scheme consists of the following algorithms.

- **Ring-sign**. A probabilistic algorithm which, on input a message m, the public keys $P_1, \ldots, P_r$ of the r ring members, together with the secret key S_s of a specific member, produces a ring signature σ for the message m.
- **Ring-verify**. A deterministic algorithm which on input (m, σ) (where σ includes the public key of all the possible signers), outputs either "`True`" or "`False`".

Properties. A ring signature scheme must satisfy the usual correctness and unforgeability properties: a fairly-generated ring signature should be accepted as valid with respect to the specified ring with overwhelming probability; and it must be infeasible for any user, except with negligible probability, to generate a valid ring signature with respect to a ring he does not belong to.

We also require the signature to be anonymous, in the sense that no verifier should be able to guess the actual signer's identity with probability greater than $1/r + \epsilon$, where r is the size of the ring, and ϵ is negligible.

Note that the size of the ring signature grows linearly with the size of the specified group: this is inherent to the notion, since the ring membership is not known in advance, and therefore, has to be provided as a part of the signature.

Combining Functions. The formal concept of a ring signature can be related to an abstract concept called *combining functions*. We slightly modified the definition given in [**?**] as follows.

Definition 1 (Combining functions). *A combining function* $C_{k,v}(y_1,\ldots,y_n)$ *takes as input a key* k*, an initialization value* v*, and a list of arbitrary values of the same length* ℓ*. It outputs a single value* $z \in \{0,1\}^\ell$*, such that for any* k, v*, any index* s *and any fixed values of* $\{y_i\}_{i\neq s}$*,* $C_{k,v}$ *is a permutation over* $\{0,1\}^\ell$*, when seen as a function of* y_s*. Moreover, this permutation is efficiently computable as well as its inverse.*

The authors of [**?**] proposed a combining function based on a symmetric encryption scheme E modeled by a (keyed) random permutation

$$z = C_{k,v}(y_1,\ldots,y_n) = E_k\Big(y_n \oplus E_k\Big(y_{n-1} \oplus E_k\big(\cdots \oplus E_k(y_1 \oplus v)\ldots\big)\Big)\Big) \quad (1)$$

For any index s, we can easily verify that $C_{k,v}$ is a combining function by rewriting equation (**??**) as follows:

$$y_s = E_k^{-1}\Big(y_{s+1} \oplus \ldots E_k^{-1}\big(y_n \oplus E_k^{-1}(z)\big)\Big) \oplus E_k\Big(y_{s-1} \oplus \ldots E_k(y_1 \oplus v)\ldots\Big)$$

2.2 Ring Signatures by Rivest *et al.* [?]

We denote by ℓ, ℓ_b, ℓ_0 three security parameters. We consider a symmetric encryption scheme E defined over $\{0,1\}^\ell$ using ℓ_0-bit keys and a hash function $\mathcal{H}$ that maps arbitrary strings on ℓ_0-bit strings. In fact, we use $\mathcal{H}$ to define the symmetric key for E. Finally, we assume that each user P_i uses a regular signature scheme built on a trapdoor one-way permutation f_i such as RSA [**?**] and that the modulus has length $\ell_b < \ell$; typically, we choose $\ell - \ell_b \geq 160$. All these assumptions follow [**?**].

The scheme proposed by Rivest, Shamir and Tauman is based on combining functions as described above. In that scheme, the inputs y_i to the combining function are computed as $f_i(x_i)$ for some $x_i \in \{0,1\}^{\ell_b}$. A ring signature on

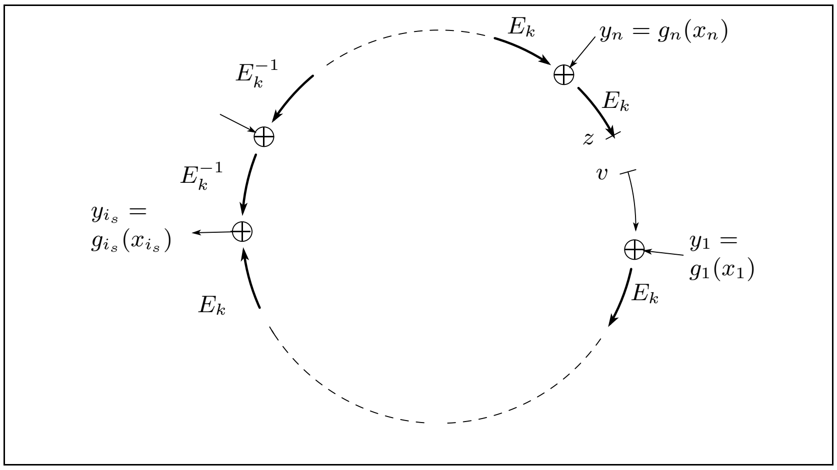

Fig. 1. The ring signature paradigm. The equation is verified if $z = v$. Given all the y_i's, the verifier just goes along the ring and checks $z \stackrel{?}{=} v$. The signer chooses v first, goes clockwise from 1 through $i_s - 1$ and counter-clockwise from n through $i_s + 1$. His trap-door allows him to extract x_{i_s}.

a message m consists in a tuple $(v, x_1, \ldots, x_n)$. Setting $z = C_{\mathcal{H}(m),v}(f_1(x_1), \ldots, f_n(x_n))$, the signature is valid iff $z = v$.

As explained in the previous section, for any message m, any fixed values v and $\{x_i\}_{i \neq s}$, one can efficiently compute the value y_s such that the combining function outputs v. Now using his knowledge of the trapdoor for function f_s, member P_s (the actual signer) is able to compute x_s such that $f_s(x_s) = y_s$. This is illustrated on Figure **??**.

However, the RSA moduli n_i involved in the scheme are different. An adaptation has to be made in order to combine them efficiently. One extends the RSA trapdoor permutation $f_i(x) = x^{e_i} \bmod n_i$ in order that all f_i's have identical domain. Briefly speaking, one defines:

$$g_i(x) = \begin{cases} q_i n_i + f_i(r_i) & \text{if } (q_i + 1) n_i \leq 2^\ell \\ x & \text{otherwise} \end{cases}$$

where $x = q_i n_i + r_i$, with $0 \leq r_i < n_i$. That is, the input x is sliced into ℓ_b-bit long parts which then go through the RSA function. The probability that a random input is unchanged becomes negligible as $\ell - \ell_b$ increases. See [**?**] for more details.

If the trap-door one-way functions are RSA functions with public exponent equal to 3, the scheme is very efficient, requiring only one modular exponentiation (and a linear number of multiplications) for signing, and only two modular multiplications per ring member for verification.

2.3 Assumptions and Security

The authors proved *unconditional* anonymity, in an information-theoretic sense. They showed that even an infinitely powerful adversary cannot guess the identity of the signer with probability greater than $1/r$. This is due to the fact that for any $k = \mathcal{H}(m)$ and any $z = v$ the equation (**??**) has exactly $(2^b)^{r-1}$ solutions which can be obtained with equal probability regardless of the signer's identity.

The unforgeability of the ring signature is based on the hardness of inverting the extended RSA permutations g_i defined above; this is easily seen equivalent to RSA assumption.

The proof provided in [**?**] holds in the ideal-cipher model. In this model, one assumes the existence of a family of keyed random permutations E_k. That is, for each parameter k, E_k and E_k^{-1} are random permutations over $\{0,1\}^\ell$; access to these permutations is modeled via *oracle queries*. The ideal-cipher model is very strong, and to the best of our knowledge it is strictly stronger than the random oracle model. See [**?**] for a discussion.

The sketch of the proof is as follows. One wants to use a forger against the ring signature scheme to invert one of the RSA permutations. That is, we are given a value $y \in \{0,1\}^\ell$ and we want to extract its ℓ-bit "cubic" root. To do so, we try to slip y as a "gap" between two consecutive E functions along the ring. Doing so, the exclusive OR between the input and the output of these E functions is set to y, and then, with non-negligible probability, the forger will have to extract the cubic root of y. Such a slip is feasible at index i only if a query "arriving" to i is asked after the other query "arriving" to or "starting" form i. The proof relies on the following lemma, which proves that this is always the case in a forgery. We refer to it as the *ring lemma*:

Lemma 1 (Ring lemma). *In any forgery output by an adversary, there must be an index 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.*

The proof provided in [**?**] is based on this lemma; one has to "guess" the index where such a situation will occur as well as the two queries involved. Thus, the guess is correct with probability at least $1/rQ_E^2$, where Q_E is the number of ideal-cipher queries. The concrete security of the scheme is related to the security of inverteing (extended) RSA by this multiplicative factor.

3 Modifications of the Existing Scheme

In this section, we explain how to significantly improve the scheme by Rivest *et al.* by removing the assumption of an ideal-cipher and obtaining at the same time a simplified proof with exactly the same security bound.

Let us first recall the *ring equation* that characterizes the verification of a ring signature:

$$\begin{aligned} v &= C_{k,v}(y_1, \ldots, y_n) \\ &= E_k\Big(y_n \oplus E_k\Big(y_{n-1} \oplus E_k\big(\cdots \oplus E_k(y_1 \oplus v) \ldots\big)\Big)\Big) \qquad \text{where } k = \mathcal{H}(m) \end{aligned}$$

A Simple Observation. The main idea consists in verifying the ring equation from another starting point than index 1 so that one just needs to go "clockwise". For instance, the signer can put an index of his choice i_0 within the signature, indicating the ring equation should start with $E_k(y_{i_0} \oplus v) \oplus \ldots$. This slight modification allows us to remove the assumption that E is a random permutation; instead, we will only need a hash function, and thus we can simply replace $E_{\mathcal{H}(m)}(x)$ by $\mathcal{H}(m, x)$.

3.1 Modified Algorithms and Simplified Security Proof

We can use this observation to simplify the original scheme. The idea is to have a signer P_s compute successive values along the ring, starting from his *own* index i_s. He chooses a random seed σ, and goes along the ring, hashing $m\|\sigma$, XOR-ing with $g_{i_s+1}(x_{i_s+1})$, concatenate with m and hash, and so on. Of course, we consider index $n+1$ as being 1. We denote the successive values as follows:

$$v_{i_s+1} = \mathcal{H}(m, \sigma), \qquad v_{i_s+2} = \mathcal{H}(m, v_{i_s+1} \oplus g_{i_s+1}(x_{i_s+1})),$$
$$\ldots\ldots \qquad v_{i_s} = \mathcal{H}(m, v_{i_s-1} \oplus g_{i_s-1}(x_{i_s-1}))$$

Just before "closing" the ring, the signer uses his secret key to extract the last input. That is, he computes x_{i_s} such that $v_{i_s} \oplus g_{i_s}(x_{i_s}) = \sigma$. Then, in order to make the signature anonymous, he chooses at random an index i_0, and outputs a modified signature $(i_0, v_{i_0}, x_1, \ldots, x_n)$. The verification is straightforward, the only point is that the verifier starts at index i_0 with value v_{i_0}. The efficiency is unchanged.

The resulting proof of unforgeability is significantly simplified, since we do not use ideal-ciphers anymore, but only a hash function. Indeed, the proof provided in [**?**] is essentially based on lemma **??**. In the original scheme, the difficulty lies in the fact that a forger can go both clockwise and counter-clockwise along the ring. When using hash functions only, the ring lemma becomes trivial: there always exist two cyclically consecutive queries such that the leading one has been asked before the trailing one. Thus, the security bound is unchanged, while the complexity assumptions are weakened.

3.2 How to Simulate a Ring

Another interesting variant in Rivest's scheme [**?**] is the following; the original scheme defines a ring signature as valid if satisfying $z = v$. However this is purely arbitrary and not necessary *as it* for the security. Indeed, such a condition can be replaced by any other condition fixing the "gap" between z and v, provided that such a "gap" cannot be chosen by a forger. For instance, one can require that $z = v \oplus E_k(0)$ instead of $z = v$.

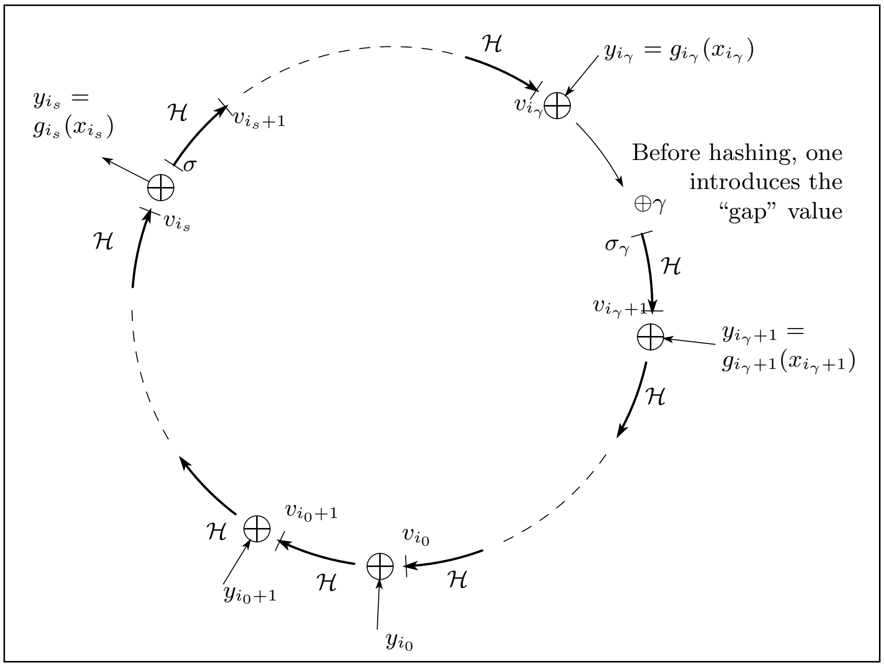

Fig. 2. The modified ring. The signer starts from its own index i_s and a seed value σ and he closes the ring when computing v_{i_s}, using his trapdoor. The verifier starts from i_0 with value v_{i_0} and checks whether the last hash outputs v_{i_0}. A "simulator" starts from i_γ and a seed σ_γ to define γ at closure only, when computing v_{i_γ}. $\mathcal{H}$ is a public hash function.

More precisely, let γ be a publicly known ℓ-bit "gap value" (for instance $\gamma = 0$ or $\gamma = E_k(0)$). We can easily produce ring signatures with only a hashing-oracle as above, but now such that γ appears as a "gap" between z and v, or, more generally, between any two consecutive indices, at any specified position i_γ. The algorithm is modified in that, when arriving at index i_γ, the verifier just replaces v_{i_γ} with $v_{i_\gamma} \oplus \gamma$ before continuing hashing and going along the ring. The figure **??** below illustrates this.

Doing so, the symmetry of the ring is somewhat broken, but in a crucial manner, since it allows us to *simulate* a ring signature, provided we can choose the value of γ freely. Viewing the gap value as a "challenge" (as in an identification scheme) the ring signature can be simulated if, and only if, the challenge can be chosen. This feature will be in crucial importance in the next section.

Modified Combining Functions. We are going to use a generalized version of the combining functions; Given a "gap" value γ, occurring at index i_γ, and a starting index i_0, we denote this modified function as $C_{v,i_0,m}$:

$$C_{v,i_0,m}(i_\gamma, \gamma, y_1, \ldots, y_n) =$$
$$\mathcal{H}(m, y_{i_0-1} \oplus \mathcal{H}(m, y_{i_0-2} \oplus \ldots \underbrace{\mathcal{H}(m, \gamma \oplus y_\gamma \oplus \mathcal{H}(\cdots \oplus \mathcal{H}(m, y_{i_0} \oplus v) \ldots))}_{v_{i_\gamma+1} \text{ is computed from } \gamma \oplus y_\gamma \oplus v_{i_\gamma}} \ldots))$$

The ring equation is verified if $C_{v,i_0,m}(.) = v$. The "gap" value being included in the arguments, this modification generalizes the definition given in [**?**]; in the original scheme, we have $i_0 = i_\gamma = 1$ and $\gamma = 0$. In the remaining of our paper, and without loss of generality, we assume that i_0 is fixed (the verifier always starts from 1), as well as i_γ (the gap appears between indices n and 1, that is $v_1 = \mathcal{H}(v_n \oplus y_n \oplus \gamma)$). We will omit i_γ in the notations and use $C_{v,i_0,m}(\gamma, y_1, \ldots, y_n)$. Also we (abusively) denote by:

$$y_s = C^{-1}_{s,\gamma,m}(\sigma_s, y_1, \ldots, y_n)$$

the solution that a signer P_s computes using seed σ_s as illustrated in Figure **??**. Keep in mind, however, that y_s must not be considered as an argument to this function.

4 Threshold and Ad-Hoc Ring Signature Schemes

In this section, we formalize the definition and security requirements for extended ring signatures, which we call *threshold ring signatures*. Traditional "t-out-of-n" threshold structures may be viewed as a special case of more general access structures, for which any criteria of minimum collaboration may be specified. In this section we also formally define such *ad-hoc groups* and the security requirements of the corresponding *ad-hoc signatures*.

4.1 Preliminaries

Assume that t users want to leak some juicy information, so that any verifier will be convinced that at least t users *among a select group* vouch for its validity. Simply constructing t ring signatures clearly does not prove that the message has been signed by different signers. A threshold ring signature scheme effectively proves that a certain minimum number of users of a certain group must have actually collaborated to produce the signature, while hiding the precise membership of the subgroup. Similarly, an ad-hoc signature might be used to modify the meaning of such a signature by giving certain members increased importance.

Definition 2. *A threshold ring signature scheme consists of two algorithms:*

- **T-ring-sign algorithm.** *On input a message m, a ring of n users including n public keys, and the secret keys of t members, it outputs a (t, n)-ring signature σ on the message m. The value of t as well as the n public keys of all concerned ring members are included in σ.*
- **T-ring-verify algorithm.** *On input a message m and a signature σ, it outputs either "*`True`*" or "*`False`*".*

We emphasize that there is no key-generation; only existing PKI is used.

The natural formalization for ad-hoc signatures follows that of threshold signatures. In both cases there are some users A_i who want to cooperate, and some others who do not cooperate, say B_i. However for the more general case, not all potential signers have equal standing. The definition of ad-hoc group captures the specification of a particular access structure. That is, it specifies which subsets of potential signers should be allowed to create a valid signature.

Definition 3. *An* **Ad-hoc group***, Σ, is a list of n users, including n certified public keys, accompanied by a list of subsets S_j of these users, called the* acceptable subsets*. This second list may be optionally replaced with a predicate defining exactly which subsets are acceptable.*

Informally, an ad-hoc signature is one which retains maximal anonymity, yet proves that the signing members all belong to at least one acceptable subset. The signature and verification algorithms are therefore relative to this structure.

Ad-hoc-sign algorithm. On input a message m, a specification of an ad-hoc group Σ (n users with some acceptable subsets), it outputs an ad-hoc-ring signature σ on the message m. The ad-hoc group structure Σ as well as the n public keys of all concerned ring members are included in σ.

Ad-hoc-verify algorithm. On input a message m and a signature σ, it outputs "`True`" if σ is a valid ad-hoc-ring-signature on m, relative to the user list and acceptable subset list specified in Σ, and "`False`" otherwise.

4.2 The Boolean Structure of Ad-Hoc Signatures.

Here we make some remarks on the structure of arbitrary ad-hoc signatures, and draw parallels with threshold ring signatures. The standard threshold ring signature construction contains all of the techniques needed for general ad-hoc signatures, and is presented in detail in Section **??**. The observations here serve to indicate exactly how the details of threshold ring signatures apply in the general case. We prefer this exposition, which provides an inclusive proof of security, yet appeals to the reader's intuition of familiar threshold structures.

A threshold ring signature scheme is clearly a special case of a general ad-hoc ring signature; out of all subgroups of the n members, every subgroup consisting of at least t members is an acceptable subset. One approach, though far from the most efficient one, is to list all acceptable subgroups and prove that one contains the cooperating signers. For each subgroup, we form the concatenation of the signatures of all concerned players; obviously such values can be simulated.

Then we use a meta-ring mechanism to prove that at least one of these $|\Sigma|$ (concatenation of) values has not been simulated but rather computed using as many private keys as needed. Indeed, such meta-ring shows that at least an acceptable subset has simultaneously received agreement and cooperation from all its members. Thus one can prove for example that player P_1 has signed m, *OR* P_2 *AND* P_3 *AND* P_4 have, *OR* P_2 *AND* P_5 have. We observe that any collection of acceptable subsets may be (recursively) described in this way, by using the boolean operations *AND, OR*.

More complex signatures may be constructed recursively. Because we have shown how to simulate a ring, and the same trivially holds for sequential composition (AND), an ad-hoc signature may serve as a node in a larger, meta-ring. For a general ad-hoc signature, a particular specification of the acceptable subsets in terms of boolean operations corresponds to a particular (nested) composite ring structure.

4.3 Our Security Model

We now derive a security model from [**?**]. We first focus on standard threshold signatures, and proceed to the general case.

Threshold Security: First of all, we want the signature to be anonymous, in the sense that no information is leaked about the group of actual signers, apart from the fact that they were at least t among the n specified ring members.

Next, we define the unforgeability property as follows. The adversary $\mathcal{A}$ is given access to a signing oracle, which can be queried to threshold-sign any message. Also, $\mathcal{A}$ can corrupt some users in order to obtain their private secret key, and is allowed to do so adaptively. Doing so, we include collusion attacks.

A t-forger is an adversary that is able to sign a message on behalf of t users, having corrupted up to $t-1$ users, under an adaptive chosen message attack. The scheme is t**-CMA**-secure (*threshold Chosen-Message Attack* -secure) if no t-forger $\mathcal{A}$ can succeed with non-negligible probability. If we denote the number of hash queries by q_H and the number of signing queries by q_S, such a probability is denoted by $\mathsf{Succ}^{\mathsf{cma}}_{t,q_H,q_S}(\ell)$, where ℓ is the security parameter.

General Ad-hoc Security: The above security definitions generalize naturally as follows. First, the scheme should be anonymous, in the sense that no information will be leaked about the group of signers except that these actual signers form an acceptable subset. For the unforgeability property, we apply the same adversarial model as above, except that $\mathcal{A}$ can corrupt any number of users, provided that no set of corrupted users contains an acceptable subset.

A Σ-forger against a signature scheme for the ad-hoc group Σ is an adversary that is able to sign a message on behalf of a Σ-acceptable subset, having corrupted no acceptable subset in its entirety, under an adaptive chosen message attack. The scheme is Σ**-CMA**-secure (*ad-hoc group Chosen-Message Attack* -secure) if no Σ-forger can succeed with non-negligible probability. Analogously to above, such a probability is denoted by $\mathsf{Succ}^{\mathsf{cma}}_{\Sigma,q_H,q_S}(\ell)$, where ℓ is the security parameter.

5 An Efficient Threshold Ring Signature Scheme

In this section, we describe a methodology to achieve the threshold ring signatures with size $\mathcal{O}(n\log n)$, while remaining essentially efficient in terms of signing/verifying. We not only provide details for signature composition, efficiency tracking, but also present a particularly efficient specification of the acceptable subsets by using fair partitions. Moreover, our solution is provably secure in the random oracle model.

Outline. Consider a ring of r members, and among them two users who want to demonstrate that they have been cooperating to produce a ring signature. The idea is to *split* the group into two disjoint sub-groups, and to show that each of these sub-groups contains one signer. However doing so may compromise perfect anonymity since such split restricts the anonymity of each user to a *sub-ring*. The solution consists in splitting the group many times, in such a way that there always exists a split for which any two users are in two different sub-rings. Then all of these splits are used as *nodes* in a *super-ring*. The super ring proves that at least one split has been solved, that is, two sub-rings has been individually solved. For the other splits, one will have to *simulate* a correct ring signature, for every unsolved sub-ring.

5.1 Fair Partitions of a Ring

Before describing our scheme, we introduce a few notations. Let t be an integer and $\pi = (\pi^1, \ldots, \pi^t)$ denote a partition of $[1, n]$ in t subsets; π defines a partition of the ring $R = (P_1, \ldots, P_n)$ in t sub-rings R^1 through R^t. Finally, the i-th bit of a string x is denoted by $[x]_i$.

Case $t = 2$. Let $\pi = \{\pi^1, \pi^2\}$ be a partition of $[1, n]$ and P_a and P_b two users that want to produce a "2-out-of-n" signature on a message m. If P_a and P_b belongs to distinct sub-rings, for instance, $P_a \in R^1$ and $P_b \in R^2$, then they are able to produce two correct ring signatures, relatively to R^1 and R^2 respectively. In that case, we say that π is a *fair partition* for indices $\{a, b\}$. If is not the case, for instance $\{P_a, P_b\} \subset R^1$, then it is infeasible for P_a and P_b to produce a valid ring signature with respect to R^2.

To ensure anonymity, we need to provide a set Π of partitions such that for any indices a and b in $[1, n]$, there exists a fair partition $\pi \in \Pi$ for $\{a, b\}$. A $(2, n)$-ring signature is a meta-ring over Π, which prove that for at least one partition π, both underlying sub-rings can be solved at the same time. Such a set can be efficiently constructed as stated in the (straightforward) following lemma:

Lemma 2. *For any integer n, there exists a set Π_n of $\lceil \log_2 n \rceil$ partitions satisfying the above requirements.*

General case. We generalize the previous definitions as follows. Let $\pi = (\pi^1, \ldots, \pi^t)$ a partition of $[1, n]$ in t subsets and $I = \{i_1, \ldots, i_t\}$ a set of t indices in $\{1, n\}$. If all integers in I belongs to t different sub-sets, for instance $i_j \in \pi^j$, we say that π is a *fair partition for* I. Intuitively, a secret is known in every sub-ring R^j defined by π.

Now, to ensure anonymity, we need to provide a set Π of partitions such that there exists a fair partition for any set of cardinality t.

Definition 4. *Let $t < n$ be two integers. We say that a set Π of partitions of $[1, n]$ is a (n, t)-complete partitioning system if for any set I of cardinality t, there exists a fair partition in Π for I:*

$$\forall I \subset [1, n],\ \#(I) = t, \quad \exists \pi = (\pi^1, \ldots, \pi^t) \in \Pi, \quad \forall j \in [1, t],\ \#(I \cap \pi^j) = 1$$

To provide such a complete system, we use the notion of *perfect hash function*. A perfect hash function for a set I is a mapping $h : [1, n] \to [1, t]$ which is 1-1 on I. A (n, t)-family of perfect hash functions, H, is such that that for any I of size t, there exists $h \in H$ which is perfect on I. It is thus clear that defining a partition in t sub-rings for each member of a (n, t)-family makes a (n, t)-complete partitioning system.

The following result has been proven in [**?**]:

Lemma 3. *There exists a (n, t)-family of perfect hash functions which has size of $2^{\mathcal{O}(t)} \log n$. Moreover each of these functions is efficiently computable.*

5.2 Description of the new Scheme

We now describe formally our scheme. We are based on the notion of fair partition, in case of a threshold scenario (which is likely to be used in practice). Consider a (n, t)-complete partitioning system and a set of t signers. If π is fair partition for this set, they can solve all the sub-rings defined by π. For the other partitions, the sub-rings are just simulated and put along a *super-ring*.

We introduce another function G, viewed as a random hash function returning $(t \times \ell)$-bit strings and we denote $p = 2^t \log n$. We assume that for all integers n and $t \leq n$, a (n, t)-complete partitioning system is publicly available : $\Pi_n = \{\pi_1, \ldots, \pi_p\}$. Finally, we introduce the straightforward notation $C_{v,i_o,m}(\gamma, y_j, j \in R)$ to properly deal with sub-rings.

Signing algorithm. We denote by $P_{i_1}, \ldots, P_{i_t}$ a subgroup of users that want to sign a message m while proving there were at least t signers among n ring members. The idea is to solve a collection of sub-ring signatures corresponding to a fair partition (each signer belongs to a sub-ring), then to concatenate the results and to apply a ring-like mechanism in order to prove that at least one of the collection of sub-rings has been entirely solved. To do so, they proceed as follows: We denote by π_s a fair partition for $I = \{i_1, \ldots, i_t\}$. We assume for simplicity that for each $j \in [1, t]$, we have $i_j \in \pi_s^j$

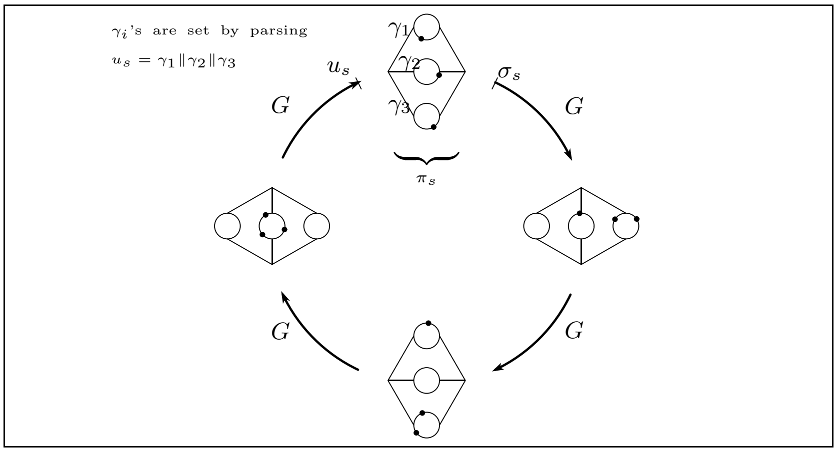

Fig. 3. The ring composition paradigm. Here we have $t = 3$. If π_s is a fair partition w.r.t. three signers, they start from a seed σ_s and compute u_s. Finally they solve the sub-rings in π_s according to the obtained gaps $\gamma_1, \dots, \gamma_3$. The equation is verified in a straightforward way, starting at any given index.

1. Choose random seeds for each sub-ring of each partition.
   `For` $i = 1, \dots, p$, `Do`
   `For` $k = 1, \dots, t$, `Do` $v_i^k \xleftarrow{R} \{0,1\}^\ell$.
2. Simulate rings for all partitions but π_s.
   `For` $i = 1, \dots, p$, $i \neq s$, `Do`
   `For` $j = 1, \dots, n$, `Do` $x_i^j \xleftarrow{R} \{0,1\}^\ell$, and $y_i^j \leftarrow g_j(x_i^j)$.
   `For` $k = 1, \dots, t$, `Do`
   $z_i^k \leftarrow C_{v_i^k,1,m}(0, y_i^j, j \in \pi_i^k(R))$ and $\gamma_i^k \leftarrow v_i^k \oplus z_i^k$.
3. Compute a "super-ring" with so-obtained gaps.
   $\sigma_s \xleftarrow{R} \{0,1\}^{t\ell}$, and $u_{s+1} \leftarrow G(\sigma_s)$
   `For` $i = s+2, \dots, p, 1, \dots, s$, `Do`
   $u_i \leftarrow G\big(u_{i-1} \oplus (\gamma_{i-1}^1 \| \dots \| \gamma_{i-1}^t)\big)$
4. Compute the gap values for sub-rings of π_s by closing the super-ring.
   $(\gamma_s^1 \| \dots \| \gamma_s^t) \leftarrow u_s \oplus \sigma_s$.
5. Solve the sub-rings for the fair partition π_s.
   `For` $j \in [1,n] \backslash I$, `Do` $x_s^j \xleftarrow{R} \{0,1\}^\ell$, and $y_s^j \leftarrow g_j(x_s^j)$.
   `For` $j \in I$, `Do`
   $\sigma_k \xleftarrow{R} \{0,1\}^\ell$ for k such that $j \in \pi_s^k$
   $y_s^j \leftarrow C^{-1}_{j,\gamma_s^k,m}(\sigma_k, y_s^j, j \in \pi_s^k(R))$ and $x_s^j \leftarrow {g_j}^{-1}(y_s^j)$.
6. Output the signature.
   $\nu \xleftarrow{R} [1,p]$ and output $\left(\nu, u_\nu, \bigcup_{1 \le i \le p} \left(x_i^1, \dots, x_i^n, v_i^1, \dots, v_i^t\right)\right)$.

Verification algorithm. A t-out-of-n ring signature is verified as follows:

1. Compute all rings starting from index 1.
   `For` $i = 1, \dots, p$, `Do`
   `For` $j = 1, \dots, n$, `Do` $y_i^j \leftarrow g_j(x_i^j)$.

For $k = 1, \ldots, t$, Do
$$z_i^k \leftarrow C_{v_i^k,1,m}(0, y_i^j, j \in \pi_i^k(R)), \text{ and } \gamma_i^k \leftarrow v_i^k \oplus z_i^k.$$

2. Verify the super-ring from index ν and obtained gaps.
$$u_\nu \stackrel{?}{=} G(\gamma_{\nu-1}^1 \| \ldots \| \gamma_{\nu-1}^t \oplus G(\ldots G(\gamma_\nu^1 \| \ldots \| \gamma_\nu^t \oplus u_\nu) \ldots)).$$

This scheme uses a super ring to prove that at least one of the partition is entirely solved. We may seen a ring (either sub-ring or super-ring) as an "OR" connective, while the partitions are used to "AND" the signatures – by concatenating the gaps $\gamma_1 \| \ldots \| \gamma_t$ before embedding them in the super-ring. Thus, our construction can be described as a composition technique.

5.3 Security result.

Theorem 1. *Our scheme is secure in the random oracle model against an adaptive chosen-message attacks involving q_H and q_G hash-queries to $\mathcal{H}$ and G respectively, and q_S signing queries, under the RSA assumption.*
$$\mathsf{Succ}^{\mathsf{cma}}_{t,\tau,q_H,q_G,q_S}(\ell) \leq q_H^2 q_G^2 t^2 \binom{n}{t} \mathsf{Succ}^{\mathsf{ow}}_{RSA}(\ell)$$

Due to lack of space, the proof will appear in the full version [?].

5.4 Discussion

We discuss here the efficiency of our threshold ring signature scheme. The size of the signature grows with both the number of users n and the number of signers t. More precisely, the size of such t-out-of-n signature is: $2^{\mathcal{O}(t)} \lceil \log_2 n \rceil \times \big(t * \ell + n * \ell\big) = \mathcal{O}(\ell 2^t n \log n)$. Signing requires t inversions of the g's functions and $\mathcal{O}(2^t n \log n)$ computations in the easy direction. This is clearly a more efficient implementation than the generic solution wherein one lists all the subgroups of cardinality t since this would lead to $\binom{n}{t} = \mathcal{O}(n^t)$ size.

This is also more efficient than the more or less straightforward solution using Shamir secret sharing scheme [?][1]. Indeed, it can be shown that such solution leads to $\mathcal{O}(n^2)$ sized signatures. This is due to additional terms that appear in the security estimates that follow from the reduction. Details will appear in the full version of the paper.

6 Conclusion

This paper addresses the open problem of allowing a subgroup of t members to sign anonymously on behalf of an ad-hoc ring. Our construction thus improves on ring signatures, group signatures, threshold signatures and on the Bellare-Rogaway paradigm for constructing composite protocols. Further work may focus on some new research by R. Canetti on universally composable protocols.

Acknowledgments. The authors thank Moni Naor and Berry Schoenmakers for helpful discussions and the anonymous referees for many extensive, detailed comments.

[1] Such solution was first suggested in preliminary version of [?].

References

1. N. Alon, R. Yuster and U. Zwick. Color Coding. J. of ACM, (42):844–856.
2. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. Expanded version of a talk given at the Nordsec '99 Workshop.
3. G. Ateniese, J. Camenisch, M. Joye, G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Crypto '00*, LNCS 1880, pp. 255–270.
4. O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system. In *PODC '01*. ACM, 2001.
5. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Eurocrypt '00*, LNCS 1807, pp. 139–155.
6. S. Brands. Untraceable off-line cash in wallets with observers. In *Crypto '93*, LNCS 773, pp. 302–318.
7. E. Bresson, J. Stern and M. Szydlo. Threshold Ring Signatures for Ad-Hoc Groups. Full version of this paper, `http://www.di.ens.fr/~bresson`.
8. J. Camenish and M. Michels. Separability and efficiency for generic group signature schemes. In *Crypto '99*, LNCS 1666, pp. 106–121.
9. J. Camenish and M. Stadler. Efficient group signatures schemes for large groups. In *Crypto '97*, LNCS 1294, pp. 410–424.
10. D. Chaum and T. Pedersen. Wallet databases with observers. In *Crypto '92*, LNCS 740, pp. 89–105.
11. D. Chaum, E. van Heyst. Group signatures. *Eurocrypt '91*, LNCS 547, pp. 257–265.
12. L. Chen and T. Pedersen. New group signature schemes. In *Eurocrypt '94*, LNCS 950, pp. 171–181.
13. R. Cramer, I. Damgård, B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Crypto '94*, LNCS 839, pp. 174–187.
14. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *Eurocrypt '96*, LNCS 1070, pp. 72–83.
15. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *FOCS '94*, pp. 454–465.
16. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto '84*, LNCS 196, pp. 10–18.
17. D. Goldschlag and S. Stubblebine. Publicly verifiable lotteries: Applications of delaying functions. In *Financial Crypto '98*, LNCS 1465, pp. 214–226.
18. Z. Haas and L. Zhou Securing Ad-Hoc Networks. In *IEEE Networks*, 13(6), 1999.
19. S. Kim, S. Park, and D. Won. Convertible group signatures. In *Asiacrypt '96*, LNCS 1163, pp. 311–321.
20. S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *ISW '97*, LNCS 1396, pp. 273–281.
21. E. Kushilevitz and T. Rabin. Fair e-lotteries and e-casinos. In *RSA Conference 2001*, LNCS 2020, pp. 100–109.
22. C. Perkins. Ad-hoc networking. Addison Wesley, 2001.
23. H. Petersen. How to convert any digital signature scheme into a group signature scheme. In *Security Protocols '97*, LNCS 1361, pp. 67–78.
24. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. of Cryptology*, 13(3):361–396, Aug. 2000.
25. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Asiacrypt '01*, LNCS 2248, pp. 552-565.
26. R. Rivest, A. Shamir, Y. Tauman. How to leak a secret. Private Com., Oct. 2001.
27. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Com. of the ACM*, 21(2):120–126, Feb. 1978.
28. A. Shamir. How to share a secret. In *Com. of the ACM*, 22(11):612–613, 1979.