Efficient Authentication for Reactive Routing Protocols

Raghav Bhaskar
Projet CODES - INRIA Rocquencourt
78153 Le Chesnay cedex, France
raghav.bhaskar@inria.fr

Javier Herranz* Fabien Laguillaumie
Projet TANC - INRIA Futurs
École Polytechnique, 91128 Palaiseau cedex, France
{herranz,laguillaumie}@lix.polytechnique.fr

Abstract

Ad hoc networks are dynamic networks formed "on the fly" by a set of nodes. Achieving secure routing in such networks is a big challenge. Asymmetric signature schemes provide mechanisms for authentication, but may result in inefficient implementations, specially when a large number of nodes is expected. Some of these efficiency problems can be mitigated with the use of aggregate signatures, which reduce the space and computations required for managing many different signatures. In this work we formalize a new concept, aggregate designated verifier signature schemes, *which is suitable for authentication of routes in reactive protocols. We propose a specific and efficient scheme with provable security in the random oracle model.*

1 Introduction

Ad hoc network is a term used to describe a network of (often) computationally constrained devices communicating over wireless channels in the absence of any central infrastructure. High dynamism (both due to physical movement of devices and members joining/leaving the network) is a typical characteristic of such networks. Finding routes to other nodes in such on-the-fly networks, in the absence of fixed routers, is a big challenge. Many protocols using varied approaches have been proposed in the literature. They can be classified into two categories: proactive and reactive routing protocols. Proactive routing protocols ([4, 5], for example) require periodic dissemination of routing information in order that all nodes are able to calculate routes to other nodes, while reactive routing protocols ([11, 6, 3], among others) are on-demand route acquisition systems wherein a node sends a route request (RREQ) whenever it needs to send a message to a node for which a route does not already exist. Reactive routing protocols are generally more scalable, since they generate less network traffic, and are thus suitable for highly dynamic ad hoc networks. But the performance of such routing protocols can be easily degraded by any malicious node present in the network. Thus it is essential to add security mechanisms to routing protocols in order to prevent and/or minimize the effect of attacks on the network. Some protocols use end-to-end authentication of routing messages using either symmetric or asymmetric cryptography, under the argument that including authentication information for each node on the route can be very expensive in terms of both computation and the size of the routing messages. But such mechanisms are still susceptible to attacks by a malicious intermediate node. Protocols using node-to-node authentication throughout the whole route are more robust but use digital signatures in the standard way, which causes them to be quite expensive. However, this efficiency drawback can be substantially mitigated with the use of *aggregate signature schemes*: many signatures on many messages from many (possibly different) signers are combined into a single shorter signature. The most efficient aggregate signature schemes proposed until now use techniques such as bilinear pairings, which result in quite costly operations when they are implemented.

Our contribution. We take a further step towards efficient authentication of reactive routing protocols, by defining the concept of *aggregate designated verifier signature scheme*. The idea is that, if a node B wants to obtain a valid (and authenticated) route to node A, only node B needs to be convinced of the authenticity of the resulting route. Thus, nodes on this route could sign their messages (and gradually aggregate these signatures) using a *designated verifier signature (DVS) scheme*: in such a scheme, a signer A sends a message in such a way that only the designated verifier B can be convinced that the message actually comes from A. In these schemes, the receiver B cannot convince anyone that the signature has been computed by A, since B could have computed this signature by himself. On the other hand, DVS can be implemented in a much more efficient way, in

*The work of this author was carried out during the tenure of an ERCIM fellowship.

terms of cost of the operations and lengths of the signatures, than standard signatures, by using, as we explicitly show in this work, *message authenticated codes* which are more efficient than asymmetric techniques.

2 Building Blocks

2.1 Aggregate Signatures

Aggregate signature schemes were introduced in 2003 by Boneh, Gentry, Lynn and Shacham [2]. Basically, aggregating signatures means compressing n signatures on n distinct messages from n distinct users into a unique (shorter) signature. This last signature is universally verifiable thanks to the signers' public keys. Thus a verifier can be provided with just one short signature rather than n, to verify whether the n users did indeed sign the n given messages. The verifier must still be provided n different public keys for verification. It is not possible to identify a misbehaving user in case of an invalid aggregate signature.

Among existing solutions, Boneh *et al.*'s proposal [2] is pairing-based and Lysyanskaya, Micali, Reyzin and Shacham' schemes [10] relies on trapdoor permutations. The required computations (modular arithmetic, pairing evaluations) can be too expensive for the constrained devices involved in an ad-hoc network.

2.2 DVS and MAC

Designated Verifier Signatures. DVS (for short) are a non-interactive variant of the designated verifier proofs introduced in 1996 by Jakobsson, Sako and Impagliazzo [7]. These non-interactive proofs have the remarkable property of being non-transferable i.e. the designated verifier accepts a signature because he knows he has not produced it by himself; but as he could have generated it on his own, he cannot convince anyone else. For the same reason, these signatures do not have one of the most important properties of traditional digital signatures, namely *non-repudiation*.

A designated verifier signature scheme is made of a common parameter generator, two key generation algorithms, a signing algorithm and a verifying algorithm. The security requirements are the following: the scheme must satisfy the property of correctness, must be existentially unforgeable under a chosen message attack, and must be *source hiding*. Source hiding means that an attacker, even if he knows all the secret keys, must be (unconditionally) unable to determine who from the signer and the designated verifier has produced a given signature.

Another anonymity property capturing the concept of *strong* designated verifier signatures of Jakobsson *et al.* [7] is *privacy of signer's identity*, studied in [8].

Message Authentication Code. MAC (for short) is a mechanism which provides mutual authentication between two users who share the same common private key. It consists of three algorithms:

$\mathsf{MAC.Key}$: it takes as input a security parameter k and returns a key space $\mathcal{K}$ and a particular key $K \in \mathcal{K}$.
$\mathsf{MAC.Gen}$: the MAC generator algorithm takes as inputs a message m and a key $K \in \mathcal{K}$, and outputs a string of bits $\tau \in \{0,1\}^\ell$ of fixed length.
$\mathsf{MAC.Verify}$: finally, the verification algorithm takes as inputs a message m, a key $K \in \mathcal{K}$ and a string of bits $\tau \in \{0,1\}^\ell$, and outputs 1 if τ is a valid MAC for m, or 0 otherwise.

With respect to security of MACs, one must prove that it is impossible to forge a valid MAC without knowing the corresponding private key K, even if the attacker has access to MAC generation and verification oracles that he can query as he wants. A bit more formally, the following game is played by a challenger and an attacker $\mathcal{A}$:

1. The challenger takes a security parameter $k \in \mathbb{N}$ and executes $K \leftarrow \mathsf{MAC.Key}(k)$. The key space $\mathcal{K}$ is given to $\mathcal{A}$.

2. The attacker $\mathcal{A}$ can make two kinds of queries:

   (i) a MAC generation query for message m_i; the challenger executes $\tau_i \leftarrow \mathsf{MAC.Gen}(m_i, K)$ and gives τ_i to $\mathcal{A}$.
   (ii) a verification query for pairs (m_i, τ_i); the challenger returns to $\mathcal{A}$ the value $\mathsf{MAC.Verify}(m_i, K, \tau_i)$.

3. At the end, the attacker outputs a pair (m, τ).

Such an attacker $\mathcal{A}$ (t', ε')-breaks the unforgeability of the MAC if it runs in total time t' and it outputs with probability ε' a pair (m, τ) such that $\mathsf{MAC.Verify}(m, K, \tau) = 1$ and the message m has not been asked as a MAC generation query in step 2(i) of the game above.

Definition 1. *A MAC is (t', ε')-secure if there is no attacker $\mathcal{A}$ which (t', ε')-breaks the unforgeability of the MAC.*

Besides this unforgeability consideration, message authentication codes also have anonymity properties. Following the terminology of [8], MAC are *source hiding*. This property is trivially obtained, as a common key is shared between two users: it is unconditionally infeasible for an attacker, even if he knows the secret key, to decide who from the two users has produced a signature.

As we will see in Section 4, a MAC can be used to construct very efficient designated verifier signatures which, moreover, can be easily and securely aggregated.

3 Aggregate Designated Verifier Signatures

In this section, we formally introduce the new concept of *aggregate designated verifier signature (Ag_DVS for short) scheme*. It consists of the following seven algorithms:

Ag_DVS.Setup: it takes as input a security parameter k and outputs the *public parameters*. These public parameters are implicit inputs of all the following algorithms;

Ag_DVS.SKeyGen: it takes as input a security parameter k and returns a pair (sk_A, pk_A) of secret and public keys for the signer;

Ag_DVS.VKeyGen: it takes as input a security parameter k and returns a pair (sk_B, pk_B) of secret and public keys for the verifier;

Ag_DVS.Sign: it takes as inputs a message m, (possibly) the public key pk_B of the designated verifier B, and the secret key sk_A of the signer A. The output is a signature σ_{AB};

Ag_DVS.Verify: the usual verification algorithm takes as inputs a message m, the public key pk_A of the signer, a signature σ_{AB} and the secret key sk_B of the designated verifier; it returns 1 if the verification is correct, and 0 if not;

Ag_DVS.Aggregate: this algorithm takes as input the public key pk_B of the verifier and n tuples[1] $\{ (pk_{A_i}, m_i, \sigma_{A_iB}) \}_{1 \leq i \leq n}$ of messages correctly signed by different users A_i for the same designated verifier B. The output is an aggregate signature Σ_B;

Ag_DVS.Ag_Verify: finally, the algorithm to verify the correctness of an aggregate designated verifier signature takes as input the list of messages and public keys $\{ (pk_{A_i}, m_i) \}_{1 \leq i \leq n}$, the aggregate signature Σ_B and the secret key sk_B of the verifier B. The output is 1 if the aggregate signature is correct and is 0 otherwise.

3.1 Security Model for Ag_DVS Schemes

We consider the highest level of security for an Ag_DVS scheme, which is obtained by combining the security models for standard designated verifier signature schemes and aggregate signature schemes. The idea is that an attacker $\mathcal{F}$ cannot obtain a valid aggregate signature for some designated verifier B, if he does not know the secret key sk_B of B, unless he knows all the secret keys sk_{A_i} of the authors of the aggregated signatures, or the aggregated signatures themselves. The attacker is allowed to make signature and verification queries. Formally, the security is defined by the following game that the attacker $\mathcal{F}$ plays against a challenger:

1. The challenger takes a security parameter k and executes Ag_DVS.Setup(k), $(sk_{A_1}, pk_{A_1}) \leftarrow$ Ag_DVS.SKeyGen(k) and $(sk_B, pk_B) \leftarrow$ Ag_DVS.VKeyGen(k).
2. The attacker $\mathcal{F}$ receives the public keys pk_{A_1} and pk_B. He can execute the protocols Ag_DVS.SKeyGen and Ag_DVS.VKeyGen by himself to obtain other pairs of secret and public keys.
3. The attacker $\mathcal{A}$ can make three kinds of queries:

   (i) a signature query for message $\tilde{m}$; the challenger executes $\tilde{\sigma}_{A_1B} \leftarrow$ Ag_DVS.Sign$(\tilde{m}, pk_B, sk_{A_1})$ and gives $\tilde{\sigma}_{A_1B}$ to $\mathcal{F}$.

   (ii) a verification query for pairs $(\tilde{m}, \tilde{\sigma}_{A_1B})$ of his choice; the challenger returns to $\mathcal{F}$ the value Ag_DVS.Verify$(\tilde{m}, pk_{A_1}, \tilde{\sigma}_{A_1B}, sk_B)$.

   (iii) a verification query for an aggregate signature $\tilde{\Sigma}_B$ corresponding to a list of pairs $\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}$ of his choice; the challenger returns to $\mathcal{F}$ the value Ag_DVS.Ag_Verify$(\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}, \tilde{\Sigma}_B, sk_B)$.
4. At the end, the attacker $\mathcal{F}$ outputs an aggregate signature Σ_B for a list of pairs $L = \{(pk_{A_i}, m_i)\}_{1 \leq i \leq n}$.

The adversary $\mathcal{F}$ succeeds if Ag_DVS.Ag_Verify$(pk_{A_1}, m_1, \ldots, pk_{A_n}, m_n, \Sigma_B, sk_B)$ $= 1$ and there is at least one pair $(pk_{A_1}, m_1) \in L$ such that m_1 has not been asked by $\mathcal{F}$ as a signature query in step 3(i) of the game above. If the running time of $\mathcal{F}$ is t and its success probability is ε, then we say that it (t, ε)-breaks the unforgeability of the Ag_DVS scheme.

Definition 2. *An Ag_DVS scheme is* (t, ε)-secure *if there is no attacker $\mathcal{F}$ which (t, ε)-breaks its unforgeability.*

The property of source hiding for standard designated verifier signature schemes can be extended to Ag_DVS.

Definition 3 (source hiding). *An Ag_DVS scheme is* source hiding *if there exists an algorithm* Simul *which takes as input only the secret key of the designated verifier and which produces bit strings which are indistinguishable (even with the knowledge of all secret keys) from the distribution of real aggregate signatures produced by real signers.*

4 Constructing Ag_DVS from MACs

The mathematical framework for the construction is the following. We consider a group $G = \langle g \rangle$ of prime order q, generated by some element g. We will suppose that the Computational Diffie-Hellman (CDH) problem is hard in this group. We explain this concept with more detail.

Definition 4. *An algorithm $(t_{CDH}, \varepsilon_{CDH})$-solves the CDH problem if it receives as input a tuple (g, g^a, g^b) for random (and secret) values $a, b \in \mathbb{Z}_q^*$, and it outputs the value g^{ab} with probability ε_{CDH} and running time t_{CDH}.*

[1] we suppose, for simplicity of the notation and w.l.o.g., that each user A_i has signed only one message m_i.

We assume that our group G is a $(t_{CDH}, \varepsilon_{CDH})$-hard group; that is, there is no algorithm which $(t_{CDH}, \varepsilon_{CDH})$-solves the CDH problem in G.

Our construction of the Ag_DVS scheme makes use of a MAC. Therefore, we will also assume that we have a (t', ε')-secure MAC, defined by the algorithms $\mathsf{MAC.Key}$, $\mathsf{MAC.Gen}$ and $\mathsf{MAC.Verify}$, with key space $\mathcal{K}$ and with fixed length ℓ for the MACs produced by $\mathsf{MAC.Gen}$ (in existing proposals of MACs, $\ell = 160$ is usually the case).

The Ag_DVS scheme that we propose consists of:

Ag_DVS.Setup: if the security parameter is k, then a $(t_{CDH}, \varepsilon_{CDH})$-hard group $G = \langle g \rangle$ is chosen such that the order q of G is a k-bit prime number. A cryptographic hash function $H : G \to \mathcal{K}$ is also chosen and published.

Ag_DVS.SKeyGen/DVS.VKeyGen: for each user U, a random number $sk_U \in \mathbb{Z}_q^*$ is chosen, and the matching public key is defined to be $pk_U = g^{sk_U}$.

Ag_DVS.Sign: given a message m, the public key pk_B of a designated verifier B, and the secret key sk_A of a signer A, the signature σ_{AB} is defined to be

$$\sigma_{AB} = \mathsf{MAC.Gen}\big(m, H(pk_B^{sk_A})\big) \ \in \{0,1\}^\ell.$$

Ag_DVS.Verify: the standard verification algorithm takes as inputs a message m, the public key pk_A of the signer, a signature σ_{AB} and the secret key sk_B of the designated verifier. It returns the bit output by

$$\mathsf{MAC.Verify}\big(m, H(pk_A^{sk_B}), \sigma_{AB}\big).$$

Note that the result of the verification is the correct one, since $pk_A^{sk_B} = pk_B^{sk_A} = g^{sk_A sk_B}$.

Ag_DVS.Aggregate: the input consists of n tuples $\{\ (pk_{A_i}, m_i, \sigma_{A_i B})\ \}_{1 \le i \le n}$. The output is the value

$$\Sigma_B = \bigoplus_{1 \le i \le n} \sigma_{A_i B} \ \in \{0,1\}^\ell.$$

Ag_DVS.Ag_Verify: this algorithm takes as input the list of messages and public keys $\{(pk_{A_i}, m_i)\}_{1 \le i \le n}$, the aggregate signature Σ_B and the secret key sk_B of the verifier B. The output is 1 if and only if

$$\Sigma_B = \bigoplus_{1 \le i \le n} \mathsf{MAC.Gen}\big(m_i, H(pk_{A_i}^{sk_B})\big).$$

Note that, roughly speaking, the verifier B must re-calculate all the aggregated signatures $\sigma_{A_i B}$ to verify the correctness of an aggregate (constant-length) signature Σ_B.

Remark 1. *The most costly operations in both signing and verifying consist of modular exponentiations (namely the computation of $pk_B^{sk_{A_i}}$ and $pk_{A_i}^{sk_B}$). These operations do not depend on the specific signed message(s), and so can be performed off-line, reducing therefore the required on-line operation to a MAC generation or verification.*

Remark 2. *In [9], Lipmaa, Wang and Bao gave a stronger model for DVS to prevent the signer and the designated verifier to delegate their signing/verifying capability without revealing their secret key. Our scheme is not secure in this model since, if A (or B) reveals $pk_B^{sk_A}$ (or $pk_A^{sk_B}$), then any user can impersonate them. Anyway, we want to stress that this situation is not relevant in our context, because if A (or B) reveals this value, he will be one of the victims of the possible attack.*

4.1 Security Analysis

We prove that the Ag_DVS scheme constructed in the previous section is secure, provided the employed MAC is secure and the CDH problem is hard in the group G. The proof, which is omitted here due to page limitation, is in the random oracle model [1] for the hash function $H : G \to \mathcal{K}$.

Theorem 1. *If the employed MAC is (t', ε')-secure and G is a $(t_{CDH}, \varepsilon_{CDH})$-hard group, then the proposed Ag_DVS scheme is (t, ε)-secure, where $\varepsilon = \varepsilon' + \varepsilon_{CDH}$ and $t = \min\{t' - Q_H, t_{CDH} - Q_H\}$, where Q_H is the total number of queries that an attacker $\mathcal{F}$ against the Ag_DVS scheme can make to the random oracle which models H.*

As a consequence, we have that the probability of breaking our scheme is $\varepsilon \le \varepsilon' + \varepsilon_{CDH}$. Therefore, the scheme is secure (small value of ε) if the employed MAC is secure (small value of ε') and the CDH problem is hard to solve in the group G (small value of ε_{CDH}).

Theorem 2. *The proposed Ag_DVS scheme is unconditionally source hiding.*

Proof. This is trivial as the designated verifier shares a key $g^{sk_B sk_{A_i}}$ with each signer A_i and therefore can produce all the MACs by himself. □

5 Application to Authenticated Reactive Routing

Reactive routing protocols enable dynamic, multi-hop routing between a set of participating nodes. They require messages to be exchanged only when a non-existing route between a source and a destination node is to be discovered or when an existing route fails. Typically three kinds of messages can be defined for reactive routing protocols: Route Request Message (RREQ), Route Reply Message (RREP) and Route Error Message (RERR).

We explain here how the above scheme can be used to secure reactive routing protocols. Consider a node B which generates a RREQ message when it wants to find a route to some node A. When the message reaches the destination A, it replies with a RREP message. Our Ag_DVS scheme enables to include authentication information from all the nodes from the source to the destination. Each node who claims to be on the route to node A (suppose that the considered route is $\{B, C_1, \ldots, C_n, A\}$) needs to provide a suitable serie of messages and an aggregate designated verifier signature on them to convince node B. In other words, node B is convinced that there exists a route from node B to node A via nodes $C_1, \ldots, C_n$ if it can obtain messages from each node saying their position in the route and an aggregate signature to confirm the authenticity of all the messages. Thus when node A generates the RREP message m_A, it also appends its signature σ_{AB} on that message, which can only be verified by node B. On receiving this RREP message, node C_n adds his own message m_n, it XORs its signature σ_{C_nB} to σ_{AB} and appends it. The same is done by nodes $C_{n-1}, \ldots, C_1$. On receiving the resulting message, node B checks for messages $m_A, m_n \ldots, m_1$ and an aggregate signature $\sigma_{AB} \bigoplus \sigma_{C_nB} \bigoplus \ldots \bigoplus \sigma_{C_1B}$ on them. What the messages actually contain depends on the used protocol. It could be the number of hops (as in AODV) or the route path (as in source routing). And also it may be possible to logically construct messages m_i from just a single message.

A drawback of the solution that we propose to secure reactive routing is that DVS do not achieve non-repudiation: a node B_1 which has found a valid and authenticated route until some node A cannot convince anyone else of this fact, because it could have produced the aggregate designated verifier signature on the corresponding RREP message by itself. Therefore, if a different node B_2 wants to obtain a valid route to A and it reaches node B_1 in this process, node B_1 cannot convince B_2 that there exists a valid route $B_1 - A$ by using its aggregate signature. The nodes in the path $B_1 - A$ must sign a message again, this time with node B_2 as designated verifier. This results in an efficiency loss if all the nodes are honest, but on the other hand it can prevent a malicious node B_1 from cheating by claiming that there exists a valid path $B_1 - A$ when it actually does not exist.

The most efficient proposal of standard aggregate signature schemes, in [2], employs bilinear pairings, whose computation is highly time consuming. With respect to the sequential schemes proposed in [10], the constructions employ certified trapdoor permutations, and result in conceptually complicated schemes, which also suffer from costly operations and long signatures (typically 1024 bits), whereas the final signatures in our scheme are no longer than 160 bits.

Therefore, it is not clear that the non-repudiation property achieved by standard aggregate signatures (with public verification) will lead to a real efficiency improvement in practice. In fact, in mobile ad-hoc networks where nodes change their position very often, the availability of routes must be almost constantly updated, and so it is difficult to imagine an “old” route $B_1 - A$ to be accepted by a node B_2 in order to construct a new route $B_2 - A$.

References

[1] M. Bellare, P. Rogaway: Random oracles are practical: a paradigm for designing efficient protocols. Proc. of 1st ACM Conference on Computer and Communications Security, 62–73 (1993)

[2] D. Boneh, C. Gentry, B. Lynn, H. Shacham: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. Proc. of Eurocrypt’03, Springer LNCS Vol. 2656, 416–432 (2003)

[3] I. Chakeres, E. Belding-Royer, C. Perkins: Dynamic MANET on-demand (DYMO) routing. IETF Internet Draft (2005)

[4] T. Clausen, P. Jacquet: Optimized link state routing protocol. RFC 3626 (2003)

[5] Y.-C. Hu, D.B. Johnson, A. Perrig: SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. Ad Hoc Networks 1 (1), 175–192 (2003)

[6] Y-C Hu, A. Perrig, D.B. Johnson: ARIADNE: a secure on-demand routing protocol for ad hoc networks. Proc. of Mobicom’02, 12–23 (2002)

[7] M. Jakobsson, K. Sako, R. Impagliazzo: Designated verifier proofs and their applications. Proc. of Eurocrypt’96, Springer LNCS Vol. 1070, 142–154 (1996)

[8] F. Laguillaumie, D. Vergnaud: Designated verifiers signature: anonymity and efficient construction from any bilinear map. Proc. of SCN’04, Springer LNCS Vol. 3352, 107–121 (2005)

[9] H. Lipmaa, G. Wang, F. Bao: Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction. Proc. of ICALP 2005, Springer LNCS Vol. 3580, 459–471 (2005)

[10] A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham: Sequential aggregate signatures from rrapdoor permutations. Proc. of Eurocrypt’04, Springer LNCS Vol. 3027, 74–90 (2004)

[11] C.E. Perkins, E.M. Royer: Ad hoc on-demand distance vector routing. Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 90–100 (1999)