Identity Based Message Authentication for Dynamic Networks

Pietro Michiardi, Refik Molva

Institut Eurecom 2229, route des Cretes BP 193 06904 Sophia-Antipolis, France
Email: {*first name.last name*}*@eurecom.fr*}

Abstract. This paper presents a message authentication scheme built on top of an original construct that combines a simple form of identity based cryptography with an iterated version of RSA. Our scheme blends the features of identity based cryptography and stream authentication while at the same time offering security comparable to that of the basic RSA cryptosystem. As opposed to other authentication schemes available in the literature, our solution does not rely on any public key infrastructure and, like any identity based cryptosystems, it does not require public key certificates. A basic security analysis, performance evaluation and storage requirements of our scheme are also provided in the paper. Furthermore, we explore a challenging application of our scheme: a scalable and lightweight key distribution service that offers authentication services to an infrastructure-less ad hoc network and that can be coupled with existing secure routing solutions.

1 Introduction

In this paper we propose a message authentication scheme (that we call IB-MAC) built on top of an original construct that combines a simple form of identity based cryptography[1] with an iterated version of RSA. In our solution, users are able to locally generate a chain of authentication material that we call *authentication tickets* using as seed the secret information (that we call a *master authentication ticket*) delivered by a key distribution center (KDC). By removing the reliance on a public key infrastructure, our scheme is particularly suitable for networks with multiple dynamic sources whereas other authentication schemes available in the literature suffer from the limitations imposed by certificate management requirements. We also describe an interesting application of our scheme: IB-MAC can be used as a basis to provide a lightweight key distribution mechanism for peer authentication in infrastructure-less ad hoc networks. In the proposed solution there is no need for a network infrastructure and the security bootstrap phase is lightweight: the key distribution center is involved neither in networking operations nor in any further security operations beyond the bootstrap phase. The remainder of the paper is organized as follows: we present the IB-MAC authentication scheme and focus on the technique used

[1] in the remainder of the paper ID-based and identity based have the same meaning

to generate the authentication material. A basic assessment of the security properties of our scheme is provided. We then focus on the performance analysis of the IB-MAC scheme both in terms of computational and message overhead and storage requirements. Finally, we describe an application of the IB-MAC scheme for key-distribution and message authentication in ad hoc networks.

2 The IB-MAC scheme

In our solution, users contact a key distribution center (KDC) and receive a secret called the *master authentication ticket* M which is tightly bound to the users' identity (ID). M is used as a seed to generate a *chain of authentication tickets* by iterative RSA encryption over the secret M. IB-MAC authentication tickets are then proposed as symmetric keying material for a message authentication protocol designed for *loosely time-synchronized users*. As opposed to similar stream authentication schemes available in the literature [9,10], our solution does not rely on any public key infrastructure and, like any ID-based cryptosystem, it does not require public key certificates for system users.
We now describe the basic stages of the IB-MAC scheme.

2.1 KDC setup

The idea behind our identity based scheme is the use of a single common RSA key pair for all users within a system. The public key is assumed to be publicly known while the private key is held by the KDC. The proposed cryptosystem uses computations in Z_n, where n is the product of two distinct odd primes p and q. For such an integer n, note that $\phi(n) = (p-1)(q-1)$. The formal description of the KDC bootstrap phase is as follows.

Key-distribution Center (KDC) setup:

1. KDC generates: two large random odd primes p and q
2. KDC computes: $n = p \cdot q \rightarrow$ RSA-like modulus (common to all users)
3. KDC selects: **small** $e \in \mathrm{Z}^*_{\phi(n)}$, $k \in \mathrm{N} \rightarrow$ Public values e and k (common to all users)
4. KDC computes: $d = e^{-k} \bmod \phi(n) \rightarrow$ Master Secret Key

The KDC uses the RSA modulus to generate a master secret key d that corresponds to a public exponent e^k: this operation is equivalent to the legacy RSA key-pair generation.
We stress that our scheme is not exposed to the well known common modulus attack whereby anyone, based on one's knowledge of a single key-pair, can simply factor the modulus and compute other users' private keys. In the present context, the secret key d is only known to the KDC and kept secret from the users of the system. Our scheme relies on a single keypair of which the private key is only known by the KDC. Further discussion on the common modulus attack is presented in section 3.

Master secret key generation As explained in section 2.1 the secret key used by the KDC to generate master authentication tickets is of the form: $d = e^{-k} \bmod \phi(n)$. Since the secret key d is generated only once during the system initialization and used to process all user requests, the KDC can afford to run a complex algorithm to generate d. However, an efficient way for calculating d can be derived based on the following observation: $d = e^{-k} \bmod \phi(n) = \left(e^{-1}\right)^k \bmod \phi(n)$. The inverse of the public exponent e can be easily calculated, and then it is sufficient to apply the square and multiply algorithm to compute the exponentiation.

2.2 Sender setup

In order to produce authenticated packets, the sender needs to contact the KDC that is in charge of issuing a master authentication ticket.

Distribution of Master Authentication Ticket:
Sender $\rightarrow$ **KDC** : ID
KDC generates: $H(ID) = C$
KDC $\rightarrow$ **Sender** : $M = C^d \bmod n$ (*securely*)

Upon verification of the sender identity ID, the KDC generates and **securely** distributes to the sender the following master authentication ticket: $M = (H(ID))^d \bmod n$, where the function $H()$ is a one-way collision resistant function, applied to the user identity ID. M can be thought of as the KDC's digital signature over the user's identity ID[2].

Sender setup:

1. Retrieve the public values n, e, k
2. Contact KDC to obtain the master authentication ticket M
3. Generate k time-dependent authentication tickets T_k

Next, the sender divides the time into uniform intervals of duration τ_{int}. Time interval 1 starts at time τ_1, time interval 2 at time $\tau_2 = \tau_1 + \tau_{int}$, etc. The sender computes authentication tickets T_i by iterative exponentiation of the master authentication ticket M using the public exponent e as shown below. Each authentication ticket is then assigned to a time interval starting with time interval τ_1 and ticket T_k, continuing with time interval τ_2 and ticket T_{k-1} and so on. The one-way authentication ticket chain is used in the reverse order of generation, so any value of a time interval can be used to derive values of previous time intervals. The sender uses the length k of the one-way chain as obtained from the KDC: this length limits the maximum transmission duration before a new one-way authentication ticket chain must be created. Note that in

[2] Note however that the public exponent used for the digital signature does not correspond to the one adopted by a legacy RSA signature

this paper we assume that chains are sufficiently long for the duration of communication. It is part of our future work to find an additional mechanism that would allow any user to self-generate a new authentication ticket chain without the need to contact the KDC [3].

Ticket generation (Sender): Generation: ↑ and Releasing: ↓ order

$T_k = M^{e^k} \bmod n$
$T_{k-1} = M^{e^{k-1}} \bmod n$
...
$T_{k-i} = M^{e^{k-i}} \bmod n, \text{ with } i \leq k$
...
$T_1 = M^e \bmod n$

2.3 Transmission of authenticated messages

Message authentication requires a source of asymmetry, such that the receivers can only verify the authentication information, but not generate valid authentication information. In our scheme we adopt the key idea behind the TESLA scheme [9] that suggest to use time as a source of asymmetry. This can be done if we assume loose time synchronized between senders and receivers: up to some time synchronization error Δ, all parties agree on the current time[4].
Subsequently to the setup phase, the sender assigns each authentication ticket sequentially to the selected time intervals (one ticket per time interval). Note that using the one-way chain of authentication tickets in reverse order of generation renders computationally infeasible for an attacker to forge authentication tickets. Furthermore, any values of a time interval can be used to derive values of previous time intervals. The sender generates a message authentication code (MAC) and attaches it to each packet. The MAC is computed over the contents of the packet that needs to be transmitted. For each packet, the sender determines the time interval and uses the corresponding value from the one-way chain of authentication tickets as a cryptographic key to compute the MAC (see [3] for details). Along with the packet, the sender also sends the authentication ticket it used to generate the MAC in the previous time interval and its unique identifier (ID). Upon receipt of a packet, the receiver verifies the authentication ticket contained in the packet and uses it to check the correctness of the

[3] In general, the need for such an additional scheme would depend on the particular scenario in which our scheme is applied. Indeed one could think of applications in which secure message authentication would be offered only to users that payed for a pre-defined amount of authentication tickets. In this case, our scheme should be extended in order to the number k of authentication tickets delivered to the users by the KDC, while at the same time preserving the basic properties of the original scheme. This is another interesting future research direction.

[4] The interested reader should refer to TESLA [9] for a thorough discussion on time synchronization issues.

MAC of the buffered packet that corresponds to the time interval of the authentication ticket. If the MAC is correct, the receiver accepts the packet. Every time a sender transmits a message, it appends a MAC to the message, using the authentication ticket corresponding to the current time interval as the key to compute the MAC. The authentication ticket for time interval τ_i remains secret until it is revealed in the packet corresponding to time interval τ_{i+1}. Figure 1 depicts the time intervals and some sample packets transmitted by the sender along time. Formally, a generic packet sent at time interval τ_i is of the form: $P_i = \{m_i, MAC_{T_{k-i}}(m_i), T_{k-(i-1)}, ID_{SENDER}\}$ where:

- m_i is the message,
- $MAC_{T_{k-i}}(m_i)$ is the MAC generated with T_{k-i},
- $T_{k-i} = M^{e^{k-i}} \bmod n$ is the ticket for time interval τ_i,
- $T_{k-(i-1)} = M^{e^{k-i-1}} \bmod n$ is the disclosed ticket for time interval τ_{i-1},
- ID_{SENDER} is the unique identifier of the sender.

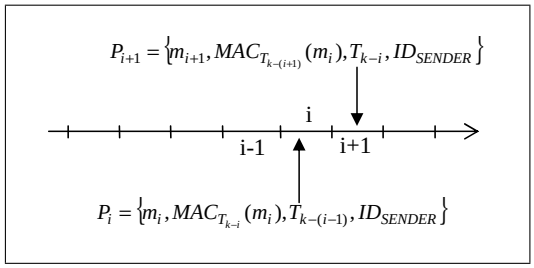

Fig. 1. Sending authenticated messages.

2.4 Verification of message authentication information at the receiver

Upon reception of packet P_{i+1} the receiver extracts the authentication ticket T_{k-i} that can be used to authenticate the previously received packet P_i. The receiver has to verify that the authentication ticket T_{k-i} corresponds to the identity ID_{SENDER} specified in the packet P_i. To that effect, the receiver only has to perform i exponentiations with e that is a small exponent[5]:

$$(T_{k-i})^{e^i} = \left(M^{e^{k-i}}\right)^{e^i} = \left(\left(C^d\right)^{e^{k-i}}\right)^{e^i} = \left(\left(C^{e^{-k}}\right)^{e^{k-i}}\right)^{e^i} = H(ID_{SENDER}) \tag{1}$$

[5] we omitted the mod$n$ notation for sake of simplicity

If $H(ID_{SENDER})$ obtained in expression (1) equals the hash function applied to the ID_{SENDER} specified in the packet P_i, then the authentication ticket is valid and it can be used as a key to verify the MAC for packet P_i.
When a sender discloses an authentication ticket, all parties potentially have access to that ticket and can create a bogus packet and forge a MAC. Therefore, as packets arrive, the receiver must also verify that their MACs are based on safe keys, i.e. a key that is only known by the sender, by checking that the time interval the sender could be in (in the example above, τ_{i+1}) is greater than the time interval corresponding to the disclosed authentication ticket (in the example above, τ_i). Receivers must discard any packet that is not safe, because it may have been forged.

3 Basic security analysis

In this section we propose a basic security analysis of the IB-MAC scheme by assuming that an attacker (internal or external) trying to break the cryptosystem is actually trying to determine the secret master key safely guarded by the key distribution center (KDC) by using disclosed authentication tickets collected over time or by performing a known-plaintext attack. Further, we consider an attacker who tries to gather a valid authentication ticket by submitting bogus identity information to the KDC or to generate valid authentication tickets from past authentication tickets. The reader should note that it is out of the scope of this section to provide a *formal* proof of the security of our scheme.

Common modulus attack: to avoid generating a different $n = p \cdot q$ modulus for each user, one could envision to fix n once and for all. A trusted central authority could provide user i with a unique pair $< e_i, d_i >$ from which user i would form a public key $< n, e_i >$ and a secret key $< n, d_i >$. At first glance, a scheme using a common modulus may seem to work: a ciphertext $C = M^{e_A} \bmod n$ intended for Alice cannot be decrypted by Bob, since Bob does not possess d_A. However the resulting system is insecure: Bob can use his own exponents $< e_B, d_B >$ to factor the modulus n. Once n is factored Bob can recover Alice's private key d_A from her public key e_A. The demonstration of how Bob can find the factorization of the common modulus n can be found in [4]. In the IB-MAC system proposed in this paper the common modulus attack is prevented. By analyzing the KDC setup phase and the sender setup phase, it is possible to observe that as compared to the typical common modulus attack scenario described above, no secret keying material is delivered to the users. The modulus n is used to generate a key d that is securely kept by the KDC. d is used to encrypt the hashed identity of a user requesting for a master authentication ticket M, unlike with the common modulus attack, and the secret M provided to each user is not a private key but the result of an encryption with the private key d. Thus, the attack detailed in [4] can not be perpetrated against the IB-MAC system.

Impersonation through blinding: suppose now that an attacker wishes to impersonate a party known under the identity ID by gaining access to the master ticket M for identity ID. The attacker knows that M is computed by encrypting the hashed identity $C = H(ID)$. Now, the attacker randomly chooses g and computes $C^* = g^{e^k} C$. Subsequently, the attacker receives the following ticket from the KDC: $M^* = (C^*)^d \bmod n$. Based on the definition of C^* we have: $M^* = (g^{e^k} C)^d \bmod n = g^{e^k \cdot d} C^d \bmod n = g \cdot M$. Thus M can be retrieved using $M = \frac{M^*}{g}$. A simple observation however shows the infeasibility of this attack: finding a bogus identifier ID^* such as $H(ID^*) = g^e C = g^e H(ID)$ requires inverting the one-way hash function $H()$, which is computationally infeasible. As a rule, the study of the impersonation attack suggests to perform the initial authentication of users applying for a master authentication ticket by requesting the full identifier ID of the user rather than a hashed value of the identifier.

Forging authentication tickets: suppose now that an attacker wishes to forge an authentication ticket using a previously revealed authentication ticket. Suppose that a legitimate sender discloses the authentication ticket: $T_k = M_{ID}^{e^k} \bmod n$, where M_{ID} is the master authentication ticket for the identifier ID. It is straightforward to show that finding M_{ID} is as hard as breaking the RSA cryptosystem. However, we want to show that also forging the authentication ticket T_{k-1} by an attacker holding T_k is as hard as breaking the RSA system. Since $T_{k-1} = M_{ID_{SENDER}}^{e^{k-1}} \bmod n = \left(M_{ID_{SENDER}}^{e^k}\right)^{e^{-1}} \bmod n$, in order to derive T_{k-1} from T_k, the attacker would have to solve the following equation: $T'_{k-1} = \sqrt[e]{T_k} \bmod n$, which is again equivalent to breaking the RSA system. On the other hand, suppose an attacker with identity ID^* holds the master authentication ticket $M^* = (C^*)^d \bmod n$. The attacker also knows $C = H(ID)$, where ID indicates the identity of a legitimate user. Let $x = \frac{C^*}{C}$. Now[6],

$$T_{k-1} = M_{ID}^{e^{k-1}} = \left(C^d\right)^{e^{k-1}} = \left(\left(\frac{C^*}{x}\right)^d\right)^{e^{k-1}} = \left(\frac{M^*}{x^d}\right)^{e^{k-1}}$$

but it is evident that the attacker cannot generate the value x^d that is needed to forge the authentication ticket T_{k-1}. Indeed: $\left(x^d\right)^{e^{k-1}} = x^{de^{k-1}} = x^{e^{-1}} = \sqrt[e]{x}$ where $d \cdot e^k = 1 \bmod \phi(n)$. Solving the e-th root of x modulo n is as hard as breaking the RSA system.

4 Performance evaluation

In this section we discuss the performance of the IB-MAC scheme in terms of computational overhead, message overhead and storage requirements. We use as a reference the TESLA scheme as it is the natural basis of IB-MAC. At first

[6] We omitted the mod$n$ notation for sake of simplicity

glance TESLA outperforms IB-MAC for the three aforementioned performance metrics. However, if we focus on an alternative performance metric that we call **bootstrap overhead**, that measure the number of messages exchanged by *all* entities when a new entity joins the group, IB-MAC shows better performances as compared to TESLA. The bootstrap cost is particularly interesting for some applications, as we will discuss in section 5.

Computational overhead: we assume the authentication ticket generation phase (as well as the TESLA key chain generation) to be executed *off-line*. In IB-MAC each ticket verification operation is equivalent to a **modular exponentiation** with exponent e (see equation 1), which is considered to be a costly operation. Ticket verification costs could be deemed prohibitive for using IB-MAC in wireless sensor networks: recent studies propose, however, the use of public-key cryptography for this type of networks [5, 6]. Conversely, a TESLA verifier only bears the cost of a **hash function execution**.
We performed some tests to assess the time needed for a verifier with limited computational power, such as mobile terminal, to verify IB-MAC authentication tickets. We studied the cost of IB-MAC ticket generation/verification for identities derived from IP addresses using a modified version of OpenSSL [1], cross-compiled for an IPAQ 38xx series with a 400Mhz X-Scale/Arm processor and Linux operating system. The choice of the hardware platform that we used for our tests is motivated by a potential application of the IB-MAC scheme that we present in section 5. Results are presented in table 1, where we assumed $k = 10000$. Note that the ticket generation time column refers to the time required by a user for the generation of k authentication tickets, expressed in seconds.

RSA key length	*Ticket generation time [s]*	*Ticket verification [ticket/s]*
512 bits	6.82	1465.8
1024 bits	19.06	524.75
2048 bits	63.57	157.3

Table 1. Performance of IB-MAC ticket generation/verification.

Message overhead: we now focus on the overhead imposed by IB-MAC for every transmitted message. Referring to figure 1, the sender needs to build a packet including the current message, the MAC of the current message and the authentication ticket used as a key for the MAC of the previous message. Each authentication ticket adds an overhead equivalent to the key size used to generate the master authentication ticket from which subsequent authentication tickets are derived. Assuming for example a key length of 512bit, each packet generated by the sender will suffer from a **64 bytes** overhead. In TESLA the message overhead depends on the hash function used to generate the TESLA keys, and can be assumed to be equivalent to 160bit, that is **20 bytes**. TESLA

saves more than 30% bandwidth as compared to IB-MAC.

Storage requirements: storage requirements can be a potential issue that has to be taken into account when designing an authentication scheme for devices with limited storage capacity. Based on a reference implementation of RSA available in the OpenSSL package, the block size of a cipher text (i.e. an authentication ticket) is equal to the key length. Using a key length of 512-bit, the authentication ticket is 512-bit long. Thus, space requirements for every mobile (sender) node to store authentication tickets is equal to: $k \cdot \text{key\_length}$, where k is the number of elements of the hash chain, i.e. the total number of authentication ticket that need to be generated, as imposed by the system parameter k. In TESLA, the sender entity has to store an hash chain of k elements, each element being of size 160bit. Again, storage requirements are less demanding than for the IB-MAC scheme. However, it should be noted that in TESLA the verifier entity stores the public key certificate of every other sender entity in the system ($N - 1$, where N is the number of entities in the group), which can be in the order of thousands of bytes [1] per certificate (depending on the key size used to sign the certificate, the length of the certificate chain, etc...). This requirement is **necessary** to verify the authenticity of the root element of the hash chain used by a potential sender.

Bootstrap overhead: bootstrapping security associations between entities represents a recurrent cost in terms of message exchange that could reduce the effectiveness of an authentication scheme such as TESLA, especially in dynamic environments in which new entities **frequently** join and leave a group. Let us consider a group of N entities that share a security association, i.e. every entity is in possession of the public key certificate of every other entity. A new member joining the group have to send her public key certificate (used to authenticate the root element of the TESLA hash chain) to every existing group member, while at the same time she should expect to receive the public key certificate of every existing member. This translates in a message exchange cost that goes as $O(N^2)$: TESLA does not scale well when the group size is large. In IB-MAC, the bootstrapping overhead is reduced to zero. IB-MAC tickets are self-authenticating since the verifier only need to know the public exponent e used by the group to verify the authenticity of a packet, as explained in section 2.4. When group joins and leave are expected to be frequent, IB-MAC represents a scalable and effective tool to bootstrap security associations with a minimum overhead.
Table 2 summarizes the different performance metrics we considered in this section. For the sake of simplicity some details have been omitted: the reader should refer to the corresponding sections to have details of overhead evaluation.

Overhead	*TESLA*	*IB-MAC*
Computation (at verifier)	1 hash function	1 modular exp.
Message	20 bytes + (N-1) certificates	64 bytes
Storage	$k \cdot 20$ bytes	$k \cdot 64$ bytes
Bootstrap	$O(N^2)$ messages	0

Table 2. Performance comparison of IB-MAC and TESLA.

5 IB-MAC for message authentication in mobile ad hoc networks (MANET)

A challenging requirement for message authentication is raised in the context of mobile ad hoc networking. As a motivating example, a variety of secure routing solutions for ad hoc networks have been proposed in the literature (see for example, [2] chapter 12). In spite of the large number of solutions, ad hoc routing does not seem to raise any new security requirement with respect to routing in classical networks, apart from key management problems that have been often left aside by current solutions available in the literature. Key management approaches try to answer the hard question of how to establish security associations with no a-priori knowledge, no a-priori trust and lack of infrastructure. Several original key management schemes based on advanced cryptographic constructs have been suggested in the literature (see [2] for a literature survey) but they all fall short in meeting the ultimate goal of building a keying infrastructure "from scratch" since they all involve a complex (and often unrealistic) key set-up phase. In this section we describe a key distribution mechanism based on the IB-MAC scheme that offers authentication services to an infrastructure-less ad hoc network. The main features of our solution range from relaxed networking infrastructure requirements to a scalable and lightweight security bootstrap phase with respect to network dynamics. The IB-MAC scheme can be coupled, for example, with the ARIADNE [7] secure routing protocol. Due to space limitation we suggest the reader to refer to [8] for a detailed description of a variation of ARIADNE based on IB-MAC.

Figure 2 shows a typical scenario in which one (or more) KDC offers both naming and authentication services. During the bootstrap phase, a mobile node (for example node N_{ID_9}) that needs authentication services contacts the closest KDC and provides initial authentication information. This initial authentication information can take the form of a secret code printed on a **prepaid card** that is delivered by a (automatic) teller, or a secret code printed on tickets delivered at the entrance of confined areas like shopping malls, airports, conference sites.

By providing the initial authentication information to the KDC, the mobile node **securely** receives a unique identifier (that in our case is represented by an IP address for the ad hoc network) and a master authentication ticket M_{ID}. Using IP addresses as node identities allows exploiting existing addressing mechanisms to provide network-wide known and unique node identifiers. However,

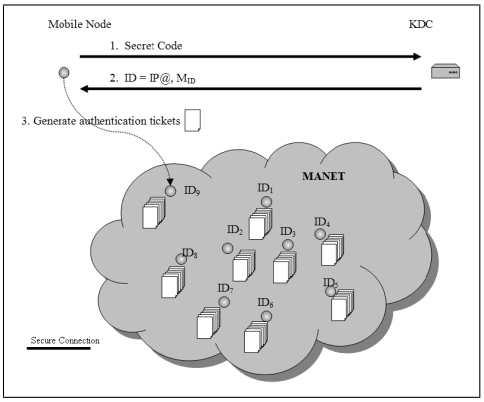

Fig. 2. Application of IB-MAC for naming and key management in open ad hoc networks.

one might consider the scenario in which nodes' IP addresses might constantly change due for example to hand overs between different ad hoc networks. Moreover, self-organized addressing schemes might be preferable to the addressing scheme proposed in this section. In these situations, an additional overhead for node re-authentication and for the generation of new authentication tickets have to be taken into account. We will address these issues in our future research, were we also plan to use cross-layer information (e.g. pseudonyms used in peer-to-peer applications) to provide a suitable naming service. The IB-MAC system is robust against impersonation through spoofing, as explained in section 3: furthermore, by introducing a monetary overhead prior to the obtention of a master authentication ticket, we make bogus authentication ticket generation an expensive operation. For the purpose of this paper we assume that authentication ticket chains are sufficiently long for the whole duration of the communication in the ad hoc network.

6 Conclusion

This paper presents an identity based authentication scheme based on a simple form of identity based cryptography combined with stream authentication techniques. In our solution, users are able to generate a chain of authentication tickets using as seed the secret information delivered by a key distribution center. By removing the reliance on a public key infrastructure, our scheme is particularly suitable for networks with multiple dynamic sources whereas other authentication schemes available in the literature suffer from the limitations imposed by certificate management requirements. In addition, there is no need for any organizational structure among users or between users and the KDC. We also provide a basic security analysis of our scheme and show through various

attacks that breaking our scheme is equivalent to breaking the basic RSA algorithm. A basic performance evaluation of IB-MAC is also provided.
Furthermore, we present an interesting application of the IB-MAC scheme in providing a lightweight and scalable key distribution service for ad hoc networks. As compared to other solutions available in the literature, our scheme trades-off an increased computational and message overhead with relaxed requirements in terms of networking infrastructure while offering a security bootstrap phase that does not entail a burdensome credential exchange between all nodes due to network dynamics.

References

1. Openssl, available from http://www.openssl.org.
2. S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile ad hoc networking.* IEEE Press, Wiley and Sons, US, 2004.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Lecture Notes in Computer Science*, 1109, 1996.
4. Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.
5. G. Gaubatz, J-P. Kaps, and B. Sunar. Public-key cryptography in sensor networks-revisited. In *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS)*, Heidelberg, Germany, August 2004.
6. V. Gligor, G. Tsudik, and D. Wagner. Security in ad-hoc and sensor networks. *Panel session in IEEE Symposium on Security and Privacy*, May 2005.
7. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom), September 2002.
8. P. Michiardi and R. Molva. Identiy based message authentication for dynamic networks. Research Report RR-pending, Institut Eurecom, 2005.
9. A. Perrig, R. Canetti, D. Tygar, and D. Song. The tesla broadcast authentication protocol. In *RSA Cryptobytes*, volume 5, 2002.
10. A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.