Institut Eurécom
2229 Route des Crêtes - BP 193
06904 Sophia-Antipolis, France

Research Report N° RR-04-111

# IDHC: ID-based Hash-Chains for broadcast authentication in wireless networks

Pietro Michiardi – Refik Molva
July 2004



| Phone: | e-Mail: |
|---|---|
| +33.4.93.00.26.45 | Piero.Michiardi@eurecom.fr |
| +33.4.93.00.26.12 | Refik.Molva@eurecom.fr |

Abstract**.** *This paper presents an authentication scheme (IDHC) based on an original concept that combines a simple form of identity-based cryptography with the Lamport's keyed hash chain method. In this solution, users contact a key distribution center (KDC) and receive a master authentication ticket M tightly bound to the users' identity. M is used as a seed to generate a chain of authentication tickets as with Lamport's keyed hash chain scheme. Our authentication scheme is designed for loosely time-synchronized users and achieves low communication and computation overhead, scales to large number of receivers, and tolerates packet loss. As opposed to other broadcast authentication schemes available in the literature, our solution does not rely on any public key infrastructure and there is no need for public key certificates. Further, there is no need for an organizational structure among users or between users and the KDC.*

*IDHC is particularly suitable for multiple and dynamic sources of broadcast traffic and we provide a challenging application of our scheme that offers peer authentication to secure the on-demand dynamic source routing (DSR) protocol for ad hoc networks.*

*A security analysis, performance evaluation and storage requirements of the IDHC scheme are also provided in the paper.*

Keywords: **id-based cryptography, authentication, wireless networks.**

## 1. INTRODUCTION

Broadcast communications have been widely addressed by the research community in order to provide reliable and efficient large-scale data dissemination networks. However, only recent studies focused on security issues that rise when considering adversarial models in which malicious users are able to impersonate legitimate users and inject fabricated packets in the network. Without an appropriate authentication scheme, malicious packet injection can be an easy task, leaving the receivers of broadcast packets with the uncertainty about the origin of the received traffic.

Broadcast authentication protocols enable the receivers to verify the identity of the source of broadcast packets. However, a simple approach based on message authentication codes (MAC) computed using a symmetric shared key between a source and every possible recipient of broadcast traffic does not guarantee the correct authentication of the source of the packets. Indeed, any malicious receiver that shares a secret key with the sender is able to forge packets and impersonate the sender. Consequently, the direction followed by broadcast authentication schemes available in the literature is based on asymmetrical cryptographic primitives that prevent this kind of attack. As a typical example, broadcast authentication schemes based on digital signatures provide a protection against sender impersonation: by signing each packet, the sender of a broadcast stream can be unambiguously authenticated. Several schemes have been proposed as optimizations of the basic digital signature approach, as signature generation and verification requires high overhead in terms of both computational power and bandwidth overhead. The overhead generated by signing every packet has been mitigated by amortizing a single signature over several packets, as proposed in [1, 2, 4, 5, 6, 7, 8].

However, none of these schemes is fully satisfactory in terms of bandwidth overhead, processing time, scalability, robustness to denial-of-service attacks, and robustness to packet loss. Furthermore, broadcast authentication schemes based on digital signatures rely on the presence of a public key infrastructure (PKI). In a typical PKI setting, a user's public key is explicitly encoded in a public key certificate that is, essentially, a binding between the certificate holder's identity and the claimed public key. The common PKI model requires universal trust in certificate issuers (Certification Authorities or CAs) and suffers from bothersome side effects such as the need for cross-domain trust and certificate revocation.

The main problem, however, is the basic assumption that all certificates are public, ubiquitous and, hence, readily available to anyone. This assumption is not always realistic, especially, in wireless (or any fault-prone) networks where connectivity is sporadic.

An alternative to digital signature schemes has been suggested in the TESLA scheme [3] where symmetric cryptographic primitives are efficiently used together with a time-synchronization protocol that provides a source of asymmetry. However, the proposed scheme relies on a public key infrastructure that has to be used in order to provide authentic "TESLA" keys thus inheriting the limitations (in terms of key management) of signature-based authentication schemes.

In this paper we propose an alternative broadcast authentication scheme built on id-based symmetric cryptographic primitives that provide a viable solution to the limitations imposed by currently available schemes. Further, we provide a "real-life" performance evaluation of the IDHC scheme executed on an X-Scale/Arm processor-based IPAQ and explore an interesting application of our authentication scheme to secure on-demand ad hoc routing protocols.

The paper is organized as follows: in section 3 we introduce the concept of identity-based cryptography and outline our assumptions used in the remaining of the paper. Section 3 provides a detailed description of the IDHC broadcast authentication scheme while in section 4 we propose a security analysis of the IDHC scheme. In section 5 we present a performance analysis of the IDHC scheme in terms of required computational power. In section 6 we present an interesting application of the IDHC scheme and in section 7 we compare IDHC to other broadcast authentication schemes available in the literature.

## 2. BACKGROUND AND ASSUMPTIONS

The idea of identity-based cryptosystem is proposed by Shamir [10] with the original motivation of simplifying certificate management in email system, thus avoiding the high cost of the public-key management and signature authentication in cryptosystems relying on a public key infrastructure (PKI). The basic idea is to find an approach in which each entity's public key can be defined by an arbitrary string. In other words, users may use some well-known information such as email addresses, IP addresses or any other unique identifier as their public key. Thus, there is no need to propagate this common information through the network. The original goal of Shamir was only partially achieved by a few solutions until the first practical identity-based encryption scheme was proposed by Boneh-Franklin [9]. Since then, several other identity-based cryptography schemes [11, 12, 13, 14, 15] have been proposed.

The common denominator of id-based cryptosystems available in the literature is that they provide the same services offered by a PKI without the management costs

of a PKI: by contacting a key distribution center (KDC), a user receives a secret key corresponding to the public key derived from the user's identity. The main issue of such systems is that the KDC possesses all secret keys corresponding to the users of the system. This limitation has been addressed and solved, for example, in [15].

In the scheme proposed in this paper, the KDC does not provide any secret keying material to the users. As it will become clear in section 3, the user receives a particular message encrypted with the secret key of the KDC that is directly related to the user's identity. In the reminder of the paper, we will refer to the message delivered to the user by the KDC as the *master authentication ticket*. The master authentication ticket is then used as a seed to generate a chain of authentication tickets as with Lamport's keyed hash chain scheme. Authentication tickets can then be compared to symmetric keys used to generate message authentication codes (MAC).

As with the general identity-based cryptosystem and PKI model, the user who wishes to obtain a master authentication ticket has to authenticate himself to the KDC. Furthermore, users' identities have to be unique and publicly available. We also assume that the KDC cannot be corrupted and that is robust to failures. A typical method to improve the KDC availability and robustness would be to distribute the KDC by using secret sharing techniques: we believe that this direction definitively needs to be explored as part of future research.

## 3. THE IDHC AUTHENTICATION PROTOCOL

A viable authentication protocol has to meet the following requirements:

- Low computation overhead for generation and verification of authentication information
- Low communication overhead
- Limited buffering required for the sender and the receiver, hence timely authentication for each individual packet
- Robustness to packet loss
- Scalability (to a large number of receivers)

The IDHC protocol meets all these requirements with low cost but it has the following special requirements:

- The sender and the receivers must be at least loosely time-synchronized
- A global and secure naming service must be available
- The receiver must buffer some messages

### *3.1. Time synchronization*

IDHC does not need the strong time synchronization properties that sophisticated time synchronization protocols provide [18, 19, 20], but only requires loose time synchronization, and that the receiver knows an upper bound on the sender's local time. A simple and secure time synchronization protocol such as the one presented in TESLA [3] can be sufficient to meet our requirements.

### *3.2. Sketch of IDHC protocol*

We first outline the main ideas behind IDHC. Broadcast authentication requires a source of asymmetry, such that the receivers can only verify the authentication information, but not generate valid authentication information. As for the TESLA protocol, IDHC uses time for asymmetry. We assume that receivers are all loosely time synchronized with the sender — up to some time synchronization error $\Delta$, all parties agree on the current time.

Here is a sketch of the basic approach:

- The sender splits up time into time intervals of uniform duration. Next, the sender generates a one-way chain of authentication tickets and assigns the values sequentially to the time intervals (one ticket per time interval). The one-way chain of authentication tickets is used in reverse order of generation so that it would be 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. 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 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.

We now describe the stages of the IDHC protocol in this order: key distribution center (KDC) setup, sender setup, sender transmission of authenticated broadcast messages, and receiver authentication of broadcast messages.

*3.3. KDC setup*

The basic idea behind the identity-based hash chain authentication protocol is the use of a single common RSA modulus *n* for all users within a system (or domain). This modulus is assumed to be publicly known.

As in RSA, 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 $e \in Z^*_{\phi(n)}$, $k \in N \rightarrow$ Public values *e* and *k* (common to all users) |
| 4. KDC computes: $d = e^{-k} \bmod \phi(n) \rightarrow$ Master Secret Key |

**Figure 1. KDC bootstrap phase.**

As sketched in Figure 1, 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 using the same modulus by multiple users in a normal RSA setting is utterly insecure. RSA is subject to a vulnerability known as the 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. However, in the present context, the secret key *d* is only known to the KDC and kept secret from the users of the system. The common modulus attack is practically impossible with our scheme, as will be discussed in more detail in section 4. We also make the important assumption that throughout the lifetime of the system, an adversary is unable to compromise a KDC.

*3.3.1. Master secret key generation*

As depicted in Figure 1, the master secret key used by the KDC to generate master authentication tickets for the users of the system 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.

*3.4. 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. Upon verification of the sender identity *ID*, the KDC generates and securely distributes to the sender the following master authentication ticket:

$$M = \left(H(ID)\right)^d \bmod n \qquad \textbf{(1)}$$

where the $H(\cdot)$ function is a one-way collision resistant function such as the popular MD5 hash function, applied to the user identity *ID*.

Expression (1) can be thought of as the KDC's digital signature over the sender identity *ID*.

We assume that the initial user authentication and the subsequent delivery of the master authentication ticket through a secure channel are performed as in the case with the registration process of a classical PKI system.

| **Sender setup:** |
|---|
| 1. Retrieve n, e, k (from a domain certificate or from the KDC) |
| 2. Contact KDC to obtain the master authentication ticket *M* |
| 3. Generate k time-dependent authentication tickets $T_k$ |

**Figure 2. Sender setup phase.**

| **Distribution of Master Authentication Ticket:** | |
|---|---|
| Sender | KDC |
| *ID* $\rightarrow$ | $H(ID) = C = h^k(M)$ |
| secretly $\leftarrow$ | $M = C^d \bmod n$ |

**Figure 3. Distribution of Master authentication ticket.**

Next, the sender divides the time into uniform intervals of duration $\tau_{\text{int}}$. Time interval 1 starts at time $\tau_1$, time interval 2 at time $\tau_2 = \tau_1 + \tau_{\text{int}}$, etc. The sender computes authentication tickets $T_i$ by subsequently

encrypting the master authentication ticket $M$ using the public exponent $e$ as shown in Figure 4. Each authentication ticket is then assigned to a time interval starting with time interval $\tau_1$ and ticket $T_k$, continuing with time interval $\tau_2$ 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[1].

**Authentication tickets generation (Sender):**

Releasing order

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

Generation order

**Figure 4. Authentication ticket generation.**

### 3.5. Broadcasting Authenticated Messages

Each authentication ticket generated using the procedure depicted in Figure 4 corresponds to a time interval. Every time a sender broadcasts 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 $\tau_i$ remains secret until it is revealed in the packet corresponding to time interval $\tau_{i+1}$.

Figure 5 depicts the time intervals and some sample packets that the sender broadcasts.

Formally, a generic packet broadcasted at time interval $\tau_i$ is of the form:

$$P_i = \{m_i, MAC_{T_i}(m_i), T_{i-1}, ID_{SENDER}\} \quad \textbf{(2)}$$

where:

- $m_i$ is the data message that the sender needs to broadcast,
- $MAC_{T_i}(m_i)$ is the message authentication code over message $m_i$ generated using the authentication ticket $T_i$ as the key,
- $T_i = M^{e^{k-i}} \bmod n$ is the authentication ticket for time interval $\tau_i$ derived from the master authentication ticket $M$ as depicted in Figure 4,
- $T_{i-1} = M^{e^{k-i-1}} \bmod n$ is the disclosed authentication ticket for time interval $\tau_{i-1}$,
- $ID_{SENDER}$ is the unique identifier of the sender.

**Broadcasting of sample authenticated packets**

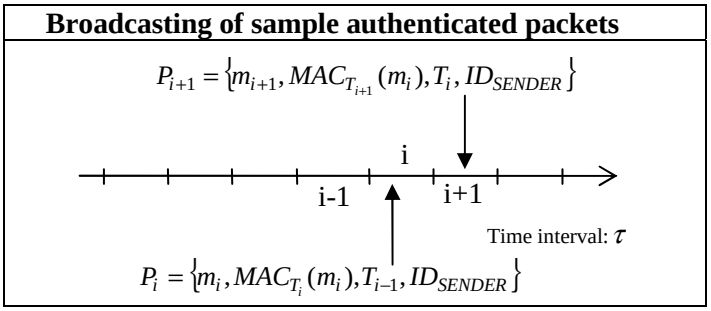


**Figure 5. Broadcasting authenticated messages.**

### 3.6. Authentication at Receiver

Upon reception of packet $P_{i+1}$ the receiver extracts the authentication ticket $T_i$ that can be used to authenticate the previously received packet $P_i$. First, the receiver has to verify that the authentication ticket $T_i$ 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:

$$(T_i)^{e^i} \bmod n = \left(M^{e^{k-i}}\right)^{e^i} \bmod n =$$
$$\left(\left(C^d\right)^{e^{k-i}}\right)^{e^i} \bmod n = \quad \textbf{(3)}$$
$$\left(\left(C^{e^{-k}}\right)^{e^{k-i}}\right)^{e^i} \bmod n =$$
$$= C^{e^{-k}e^{k-i}e^i} \bmod n = C = H(ID_{SENDER})$$

If $H(ID_{SENDER})$ obtained in expression (3) 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, $\tau_{i+1}$) is greater than the time interval corresponding to the disclosed authentication ticket (in the example above, $\tau_i$). Receivers must discard any packet that is not safe, because it may have been forged.

[1] For this article we assume that chains are sufficiently long for the duration of communication.

## 4. SECURITY ANALYSIS

In this section we propose a security analysis of the IDHC 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.

In section 4.5 we discuss about the choice of the system parameter *k* that avoids duplicate authentication ticket generation and prevents the re-use of past authentication tickets by an attacker.

### *4.1. Common modulus attack*

In a naïve setting of RSA-based cryptosystem, to avoid generating a different modulus $n = p \cdot q$ for each user, one could envision to fix $n$ once and for all. The same $n$ could then be used by all users. 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 $< n, d_i >$.

However, an observation due to Simmons shows that an RSA modulus should never be used by more than one entity. Indeed, 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, this is incorrect, and 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 [23].

In the IDHC system proposed in this paper, however, the common modulus attack is prevented even if all entities of the systems share a common modulus. By carefully analyzing the KDC setup phase (section 3.3) and the sender setup phase (section 3.4), 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. Instead, the common modulus $n$ is used to generate a master secret key $d$ that is securely kept by the KDC. The key $d$ is used to encrypt the hashed identity of the 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 [23] can not be perpetrated against the IDHC system.

### *4.2. Impersonation through blinding*

Suppose now that an attacker wishes to impersonate a party known under the identity *ID* by maliciously gaining access to the master authentication ticket $M$ for identity $ID$.

The attacker knows that the master ticket $M$ is computed by the KDC 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 master authentication 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 $H(\cdot)$, 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.

### *4.3. Forging authentication tickets*

In this section we suppose that an attacker wishes to forge an authentication ticket by using a previously revealed valid 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,

$$T_{k-1} = M_{ID}^{e^{k-1}} \bmod n = \left(C^d\right)^{e^{k-1}} \bmod n =$$

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

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:

$$\left(x^d\right)^{e^{k-1}} \bmod n = x^{de^{k-1}} \bmod n = x^{e^{-1}} \bmod n = \sqrt[e]{x} \bmod n$$

where $d \cdot e^k = 1 \bmod \phi(n)$

Again, solving the *e*-th root of *x* modulo *n* is as hard as breaking the RSA system.

### *4.4. Known-plaintext attack*

We want now to examine another kind of elementary attack that could be perpetrated by an attacker wishing to determine the secret key *d* used by the KDC to generate master authentication tickets. The known-plaintext attack is a form of cryptanalysis where the attacker knows both the plaintext and the associated ciphertext. In the IDHC context, an attacker is able to determine the plaintext associated to every master authentication ticket since the KDC generates it from the public identity of the requesting user. However, the master authentication ticket is delivered in a secure way to the corresponding user, which in turn only reveals authentication tickets generated as in section 3.4. An attacker needs to know the secret key *d* in order to extract the master authentication ticket.

It is worth mentioning that the operation carried out by the KDC when delivering master authentication tickets to the users of the system is comparable to the generation of a (RSA) digital signature on a message (in our case, the hashed identity of a user) using the secret key *d*. Thus, it is possible to affirm that the IDHC system is as secure as the (RSA) digital signature scheme.

### *4.5. Choice of system parameter k*

The parameter *k* determines the number of authentication tickets that can be generated by the user. However, *k* cannot take on any arbitrary value. A simple observation is sufficient to characterize the choice of *k*. By construction (see Figure 4) an authentication ticket takes the following expression: $T_k = M^{e^k} \bmod n$. Now, we would have to find an integer $m \neq k$, such as:

$$M^{e^k} \bmod n = M^{e^m} \bmod n, \text{ with } m \neq k \quad \textbf{(4)}$$

It is trivial to show that $m = k \bmod \phi(\phi(n))$, so as long as $k < \phi(\phi(n))$ the following implication holds:

$$M^{e^k} \bmod n = M^{e^m} \bmod n \Rightarrow m = k \quad \textbf{(5)}$$

By choosing $k < \phi(\phi(n))$ we avoid duplicate authentication tickets.

## 5. PERFORMANCE ANALYSIS

When plain RSA is used for encryption, the public encryption exponent *e* is typically a small integer with only a few 1-bits. One example is the popular OpenSSL toolkit [22] that uses 65537 as the default public key value for RSA certificates. Encryption with such small exponents can be accelerated with specialized algorithms for modular exponentiation. In our setting, the secret/public key generation phase is equivalent to an RSA key generation while the master authentication ticket generation (performed by the key distribution server) can be considered equivalent to a RSA signature over the public identity of the requesting node. However, it is critical to evaluate the computation power requirements that a user (a mobile node of the network) has to satisfy in order to generate authentication tickets. Even by choosing a relatively small exponent *e*, single nodes have to deal with the generation of *k* authentication tickets, an operation that can be compared to *k* RSA encryptions. Finally, the verification performed at the receivers is equivalent to an RSA-signature verification. We ran some simple tests to assess the cost of IDHC authentication ticket generation/verification for public keys derived from IP addresses. The encryption and verification was tested using OpenSSL cross-compiled for an IPAQ 38xx series with a 400Mhz X-Scale/Arm processor and Linux Familiar operating system [21, 22]. Results are presented in Table 1.

| RSA | Generation [ticket/s] | Verification [ticket/s] |
|---|---|---|
| 512 bits | 121.48 | 1475.8 |
| 1024 bits | 26.87 | 524.75 |
| 2048 bits | 4.61 | 157.3 |
| 4096 bits | 0.7 | 47.58 |

**Table 1. Performance comparison of IDHC Ticket generation/verification with different key-lengths.**

Taking as an example an RSA key length of 512 bits, a node can generate 121.48 authentication tickets per

second while a potential receiver is able to authenticate 1475.8 packets per second. Results gain more relevance when the IDHC scheme is applied to a specific scenario. In section 6 we present a potential application of the IDHC scheme to secure reactive routing protocols for ad hoc networks and assess the viability of the IDHC scheme.

### 5.1. *Storage requirements*

If computational power requirements are satisfied by the IDHC authentication scheme, also storage requirements can be a potential issue that has to be taken into account when designing an authentication scheme for mobile devices which have a limited storage capacity. Based on a reference implementation of RSA available in the OpenSSL package, it is straightforward to evaluate the space requirements for a single authentication ticket that needs to be stored in every node of the network. Indeed, the block size of a cipher text (i.e. an authentication ticket) generated as depicted in Figure 4 is equal to the key length. For example, by taking a key length of 512-bit, also the authentication ticket would be 512-bit long. To be more precise: the *ID* used to generate the master authentication ticket consists of 32-bit, since we used as unique identifiers IPv4 addresses. The popular hash function MD5 applied to the *ID* results in a 128-bit message digest. The master authentication ticket generated by the KDC for identity *ID* will be as long as the key length used to generate it: for example using 512-bit ID's authentication tickets generated by the mobile nodes would also be 512-bit long.

Thus, space requirements for every mobile node is equal to: $k \cdot key\_length$, 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*.

## 6. APPLICATION

A particularly challenging requirement for peer authentication is raised by secure routing protocols in the context of mobile ad hoc networking. In this section we outline a lightweight key distribution scheme based on our solution that offers an authentication service to an infrastructure-less ad hoc network. The main features of the proposed solution are:

- There is no need for a network infrastructure
- The security bootstrap phase is lightweight and node-oriented as opposed to network-oriented[2]
- There is no need for an organizational structure among peers or between peers and the key distribution center (KDC)
- The KDC is not involved in networking operations
- The KDC is not involved in any further security operations beyond the bootstrap phase

### 6.1. *Sketch of the proposed solution*

Figure 6 represents a typical scenario in which one (or more) KDC offers both naming and authentication services. During the security bootstrap phase, prior to joining the ad hoc network (which might already exists), a mobile node that needs authentication services has to contact the closest KDC and provide some 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
- a secret code printed on tickets delivered at the entrance of confined areas like malls, airports, conference sites
- etc…

The node has then to initiate a secure socket layer (SSL) connection with the KDC server. The SSL connection provides server authentication and a secure channel that is used for subsequent communications. By providing the initial authentication information to the KDC, the mobile node receives a unique identifier (that in our case is represented by an IP address for the ad hoc network) and a master authentication ticket generated by the KDC for the delivered identity as explained in section 3.4 and in Figure 3.

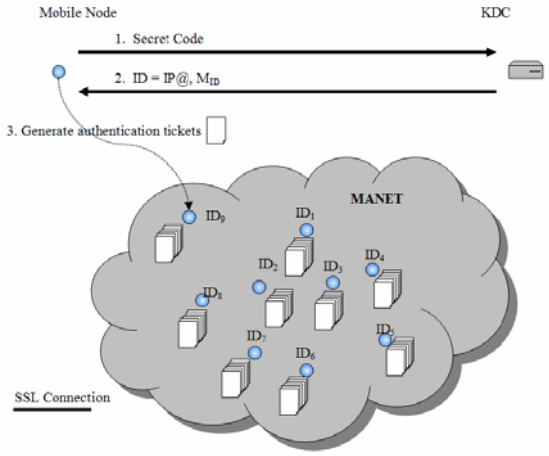


**Figure 6. Ad hoc network scenario: security infrastructure**

Like the TESLA protocol, the IDHC broadcast authentication scheme is particularly suitable to secure on-demand routing protocols for ad hoc networks. In [16] the authors propose to use a slight variation of TESLA called "ARIADNE" in order to authenticate the

[2] In general, the literature offers key distribution schemes in which all nodes have to be initialized at the same time of network creation. In our case the initialization phase is only performed for the node joining the network.

route discovery phase of the dynamic source routing (DSR) protocol. In this section, we propose a variation of the "ARIADNE" protocol that is based on the IDHC authentication scheme. Due to lack of space, we only discuss about the impact of using the IDHC scheme in an ad hoc setting rather than providing the details of the secure routing protocol. Here we emphasize that instead of using TESLA keys to generate a keyed hash chain used for packet authentication, routing messages are authenticated by using IDHC authentication tickets as illustrated in Figure 5.

By adopting the IDHC authentication scheme, key management requirements are significantly reduced with respect to the original TESLA-based protocol. Indeed, as explained in section 7, the TESLA scheme must rely on a public key infrastructure (PKI). However, as opposed to a classical PKI client that must have a valid public key certificate of the certification authority that issued all the certificates for the other users, in our scheme the identity of another peer can be verified without the need of a public key certificate. In addition, key revocation is greatly simplified with respect to a classical PKI system: authentication tickets are limited in number (only $k$ tickets) and their validity can be made limited in time or in utilization, by simply appending a validity period or authorization information to the identity used to generate the master authentication ticket.

Furthermore, in our scheme there is no need for an organizational infrastructure among peers, which can be operated by entities belonging to different organizations. The KDC is not involved in any networking operations, i.e. it must not be on-line during the network operation, and is not involved in any further security operations other than the bootstrap phase or when the authentication ticket pool is exhausted. Furthermore, the initial peer authentication to the KDC is only needed once. Indeed, the mobile node can contact the KDC to renew the master authentication ticket by presenting the last self-generated hash chain element/authentication ticket.

The viability of the IDHC authentication scheme can be assessed by taking the values reported in Table 1 and comparing them with the average number of control packets sent by all the nodes of an ad hoc network to discover and maintain routes. In [17] the authors provide a simulation-based study of the control overhead generated by three ad hoc routing protocols. Specifically, for the DSR protocol the average control traffic for a typical scenario with 40 mobile nodes in a 4km by 4km area and 20 CBR data flows consists of 3000 packets during all the simulation period (900 seconds). Thus, in average, every node generates five control packets per minute. Obviously, this value can be relatively higher if we consider critical scenarios with high mobility or dense traffic but for an approximate evaluation of the IDHC authentication scheme it is believe sufficient to take an average value. The generation and verification rates of authentication tickets reported in Table 1 are sufficiently high to support the average control traffic generated by nodes in the simulation scenarios presented in [17] leading to the conclusion that the IDHC scheme is an effective solution to secure the DSR routing protocol.

## 7. RELATED WORK

The TESLA broadcast authentication protocol [3] has been used as a basis for the design of the IDHC protocol. The fundamental idea behind the TESLA scheme is that time is used as a source of asymmetry while using symmetric cryptographic primitives in order to maintain a low computational overhead. In TESLA, the source of broadcast traffic splits up time in uniform intervals and generates a hash chain of length $k$. Hash chain elements are then used in reverse order and serve as keys for the generation of a keyed message authentication code (MAC) that is appended to each transmitted packet. The verifier needs to wait a predefined time interval to retrieve the key that has been used to generate the MAC. Only keys that have been correctly revealed by the source, which is loosely synchronized with the receivers, can be used to validate or discard a received packet. The main drawback of the TESLA authentication protocol is that revealing hash chain elements does not guarantee a proper authentication of the sender. Indeed, the root of the TESLA hash chain needs to be certified by a universally trusted third party (a certification authority for example) in order to be sure that all the hash chain elements belong to the sender with identity *ID*. Precisely, the hash chain root has to be digitally signed with a secret key belonging to user known under the identity *ID*. The corresponding public key has to be certified by a certification authority that guarantees the binding between the private/public key pair and the identity *ID*. A potential receiver has to validate (only once) that the root of the hash chain belongs to the sender that is generating the broadcast traffic. This requirement however implies the reliance on some public key infrastructure (PKI) for both certificate generation and revocation.

The key idea behind the authentication scheme presented in this paper is that the IDHC scheme preserves the main advantages of the TESLA scheme but does not rely on any PKI. Indeed, by applying the fundamental principles of id-based cryptosystems, an authentication ticket that is used for packet authentication is directly related to the identity *ID* of the source of the broadcast traffic and there is no need for an on-line certification authority. The price to pay for such

a simplification in the key management requirements is that the cryptographic primitives used in the IDHC scheme are no longer symmetric. Both storage requirements and computational power are moderately higher than in the TESLA scheme. As an example, a TESLA hash chain element requires 128-bit of space (if the MD5 algorithm is used as hashing function) while an IDHC authentication ticket depends on the key-length used to generate it (typically 512-bit). However, as compared to signature-based authentication schemes, the essential advantage of the IDHC mechanism is that authentication tickets can be *pre-computed* and their verification is fast due to the small exponent *e*.

## 8. CONCLUSION AND FUTURE WORK

This paper presents an authentication scheme (IDHC) based on an original concept that combines a simple form of identity-based cryptography with the Lamport's keyed hash chain method. In our solution, users are able to generate a chain of authentication tickets using as seed the secret information (*i.e.* the 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. In addition, there is no need for any organizational structure among users or between users and the KDC.

Our broadcast authentication scheme is designed for loosely time-synchronized users and achieves low communication and computational overhead, scales to large numbers of receivers, and tolerates packet loss.

We also provide a detailed security analysis of our scheme and show through various attacks that breaking our scheme is equivalent to breaking the basic RSA algorithm.

The viability of the IDHC scheme is verified through a performance analysis of our solution, as well as an evaluation of storage requirements. Our implementation is based on the OpenSSL package and has been cross-compiled to be executed on the ARM/X-Scale platforms such as the IPAQ 38xx series.

Furthermore, we present an interesting application of the IDHC scheme. We provide a lightweight key distribution service that offers peer authentication to an infrastructure-less ad hoc network. In our scheme, there is no need for a network infrastructure and the security bootstrap phase is lightweight. Further, the key distribution center is involved neither in networking operations nor in any further security operations beyond the bootstrap phase.

## 9. REFERENCES


[1] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, Efficient and secure source authentication for multicast, in Proceedings of NDSS 2001

[2] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, Efficient authentication and signature of multicast streams over lossy channels, in Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 56–73, May 2000.

[3] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, The TESLA Broadcast authentication Protocol, in RSA Cryptobytes 2002

[4] R. Gennaro and P. Rohatgi. How to sign digital streams. In Advances in Cryptology — CRYPTO '97, volume 1294 of Lecture Notes in Computer Science, pages 180–197, 1997.

[5] S. Miner and J. Staddon. Graph-based authentication of digital streams. In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 232–246, May 2001.

[6] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In Proceedings of the 6th ACM CCS, pages 93–100, November 1999.

[7] D. Song, D. Zuckerman, and J. D. Tygar. Expander graphs for digital stream authentication and robust overlay networks. In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 258–270, May 2002.

[8] C.Wong and S. Lam. Digital signatures for flows and multicasts. In IEEE ICNP '98, 1998.

[9] D. Boneh, M. Franklin, Identity based encryption from the Weil Pairing, in Advances in Cryptography, CRYPTO 2001

[10] A. Shamir, Identity based cryptosystems and signature schemes, in Proceedings of Advances in Cryptology, 1984

[11] C. Cocks, An identity based encryption scheme based on quadratic residues, in B. Honary (Ed.), Cryptography and Coding, Lecture Notes in Computer Science, vol. 2260, Springer, Berlin, 2001

[12] F. Hess, Efficient identity based signature schemes based on pairings, in Proceedings of the 9th Workshop SAC 2002

[13] J.C. Cha, J.H. Cheon, An identity based signature from Gap Diffie-Hellman groups, Cryptology ePrint Archive, Report 2002/018

[14] K.G. Paterson, ID-based signatures from pairing on elliptic curves, Cryptology ePrint Archive, Report 2002/004.

[15] X.Ding, G. Tsudik, Simple Identity-based Encryption with Mediated RSA, RSA Conference 2003, Cryptographer's Track (CT-RSA'03), San Francisco, 2003

[16] Y-C. Hu, A. Perrig, D. B. Johnson. Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks, in Proceedings of MobiCom 2002, September 23-28, 2002, Atlanta, Georgia, USA

[17] H. Jiang, J. J. Garcia-Luna-Aceves, Performance comparison of three routing protocols for ad hoc networks, in Proceedings of ICCCN2001.

[18] L. Lamport and P. Melliar-Smith, Synchronizing clocks in the presence of faults, Journal of the ACM, 32(1):52–78, 1985.

[19] D. Mills, Network Time Protocol (version 3) specification, implementation and analysis, Internet Request for Comment RFC 1305, Internet Engineering Task Force, March 1992.

[20] B. Simons, J. Lundelius-Welch, and N. Lynch, An overview of clock synchronization, In B. Simons and A. Spector, editors, Fault-Tolerant Distributed Computing, number 448 in LNCS, pages 84–96, 1990.

[21] Linux Familiar Distribution, available from http://www.handhelds.org

[22] OpenSSL, available from http://www.openssl.org

[23] D. Boneh, Twenty years of attacks on the RSA Cryptosystem, in Notices of the AMS, February 1999