SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks

Srdjan Čapkun[1] Levente Buttyán[2] Jean-Pierre Hubaux[3]

[1,3]Laboratory for Computer Communications and Applications (LCA)
Swiss Federal Institute of Technology Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
srdan.capkun@epfl.ch, jean-pierre.hubaux@epfl.ch

[2]Budapest University of Technology and Economics
Department of Telecommunications
Magyar tudosok krt. 2
H-1117 Budapest, Hungary
buttyan@hit.bme.hu

ABSTRACT

In this paper we present **SECTOR**, a set of mechanisms for the secure verification of the time of encounters between nodes in multi-hop wireless networks. This information can be used notably to prevent wormhole attacks (without requiring any clock synchronization), to secure routing protocols based on last encounters (with only loose clock synchronization), and to control the topology of the network. SECTOR is based primarily on distance-bounding techniques, on one-way hash chains and on Merkle hash trees. We analyze the communication, computation and storage complexity of the proposed mechanisms and we show that, due to their efficiency and simplicity, they are compliant with the limited resources of most mobile devices.[1]

Categories and Subject Descriptors

C.0 [**Computer-Communication Networks**]: [Security and protection]

General Terms

Security, Positioning

Keywords

Security, Mobile Networks, Mobility, Security associations, Positioning

[1]The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 (`http://www.terminodes.org`).

ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 31, 2003, Washington, USA.

1. INTRODUCTION

In multi-hop wireless networks[2], keeping track of node encounters is a crucial function, to which the research community has devoted very little attention so far. This function can be used for the detection of wormhole attacks, to secure routing protocols based on the history of encounters, and for the detection of cheating attempts (e.g, in charging mechanisms).

In this paper, we propose a set of mechanisms for the *secure verification* of the time of encounters between nodes in multi-hop wireless networks. Our proposal enables any node to prove to any other node (or base station) its encounters with other nodes before or at some specific time; we call it SECTOR (SECure Tracking Of node encounteRs).

SECTOR can be used to **prevent wormhole attacks** [12, 20, 24] in ad hoc networks, without requiring any clock synchronization or location information; it is therefore a valid alternative to the other solutions already proposed to this problem.

SECTOR can also help to **secure routing protocols** in mobile ad hoc networks, which are based on the history of encounters; we illustrate this with FRESH [10], the last-encounter protocol that enables an efficient route discovery for large-scale ad hoc networks.

In addition, if it is applied in a multi-hop cellular network, SECTOR allows a base station to partially or totally reconstruct the network topology, in real time, or at some past time, in a secure way. This information can then, for example, be used by the network operator for the **detection of node misbehavior**; it can also help a base station to securely position the nodes located out of its power range. An example of the latter application is the prevention of charging frauds in multi-hop cellular networks [2]. If charging is based on a probabilistic micro-payment scheme [17],

[2]By multi-hop wireless networks, we mean those networks in which communication is partially or totally relayed by several mobile nodes; multi-hop wireless networks include "pure" or "autonomous" ad hoc networks, ad hoc networks that have sporadic access to a backbone, and multi-hop cellular networks

the proposed mechanism can help the operator to identify inconsistencies that may correspond to fraud attempts.

A last example of application of SECTOR is to use it for **topology monitoring** in mobile ad hoc networks: the secure knowledge of node encounters can help detecting attackers that use multiple nodes while assigning them the same identity.

Several research efforts that have been reported propose various location verification mechanisms for mobile networks (some of them are extremely recent). Waters and Felten [27] propose a system for proving the location of tamper-resistant devices, based on the exchange of RF messages. The system uses round-trip time of flight measurements to distance-bound the devices. A similar protocol, based both on RF and ultrasound, is devised by Sastry, Shankar and Wagner [25]. Finally, Brands and Chaum [4] have proposed a set of efficient distance-bounding protocols that operate with bit exchange and rely on the measurements of round-trip time of flight.

In the area of ad hoc networks, Stajano [26] recommends the use of location-limited channels to provide authentication in ad hoc networks. Balfanz et al. [1] make use of location-limited channels for location based access control. It has to be stressed that these research efforts aim at securing information related to *location*, whereas SECTOR deals with *node encounters*.

When engineering SECTOR, we carefully took the limited memory and computational resources of mobile devices into account. Thus, the proposed mechanisms are based primarily on one-way hash chains and Merkle hash trees [19]. We discuss how public-key cryptography could be used; considering its dramatic computational cost, we show that its use can be avoided.

One-way hash chains and Merkle hash trees have already been used to secure various aspects of routing. Hauser, Przygienda and Tsudik [11] present an efficient mechanism for the authentication of link state routing updates. Zhang [28] improves this mechanism and presents a chained Merkle-Witnernitz one-time signature. Hu, Perrig and Johnson [15] propose a set of efficient security mechanisms for routing protocols, which make use of hash chains and Merkle hash trees. They also use hash chains to efficiently secure distance vector routing updates in SEAD [13] and to prevent malicious changes of hop count in Ariadne [14].

The mechanisms of SECTOR differ in their complexity and can be adapted to the security requirements as well as to the organization of the network, such as the existence of on-line or off-line central authorities. We will analyze the communication, computation and storage complexity of the proposed mechanisms and we will show that, due to their efficiency and simplicity, they can be easily integrated in a variety of multi-hop wireless networks.

This work has been carried out in the framework of the Terminodes Project [16].

The organization of this paper is the following. In Section 2, we describe the system model and the assumptions. In Section 3, we explain the mechanisms we propose to support the secure verification of encounters. Sections 4 and 5 contain the security and performance analysis. In Section 6 we describe several potential applications. We conclude the paper in Section 7.

2. SYSTEM MODEL AND ASSUMPTIONS

Before we describe the mechanisms, we shortly describe our system model and the assumptions.

Our system consists of a set of mobile nodes and it may also contain a set of (fixed) base stations. Nodes communicate using radio transmissions. If two nodes reside within the power range of each other, then they are considered to be neighbors. We assume that the radio link between neighbors is bidirectional. We do not make any specific assumptions about the medium access control protocol used by the nodes to access the radio channel.

The nodes may form a pure ad hoc network, an ad hoc network that has sporadic access to a backbone, or a multi-hop cellular network. In all cases, communication between distant parties may involve multiple wireless hops. We do not make any specific assumptions about the routing protocol used to transfer packets from their source to their destination.

Each node has a local clock, and we assume that the clocks of the nodes are loosely synchronized. By this we mean that the difference between the clocks of any two nodes is typically smaller than 1 second. How loose time synchronization is achieved is out of the scope of this paper. We note however that some proposals have been developed [23].

We assume that the nodes can measure time locally with nanosecond precision. One of our proposed protocols (MAD) also assumes that each node is equipped with a special hardware module that can temporarily take over the control of the radio transceiver unit of the node from the CPU. We assume that with the help of this special hardware module, the node can receive a single bit, perform a XOR operation on two bits, and then transmit a single bit without involving the CPU of the node. In other words, we assume that the node can be put in a special state where it is capable of responding to a one-bit challenge with a one-bit response essentially immediately, without the delay imposed by the usual way of processing messages. We assume that the bits are correctly transmitted, meaning that there are no collisions and no jamming. If collisions occur or if jamming is detected, the protocol is terminated by the party that detects it.

We do not assume that the nodes are equipped with positioning devices, nor that they can obtain their geographical locations in any other way. We only assume that each node is able to generate cryptographic keys, to check signatures, to compute hash functions, and more generally, to accomplish any task required to secure its communications (including to agree on cryptographic protocols with other nodes).

We assume that the network operates with a central authority. This authority can be on-line, meaning that the author-

ity operates on-line servers that can be contacted using the network (by single hop or multi-hop communication), or off-line, meaning that the services of the authority cannot be reached via the network. In any case, the authority controls the network membership and assigns a unique identity to each node.

We assume that all network nodes either share pairwise secret keys, or hold each others' authentic public keys. This can be achieved by manually pre-loading all keys into the nodes in a network setup phase, however, this approach is inflexible and can prevent the introduction of new nodes in the network. Pairwise secret keys can also be established by using a probabilistic key distribution scheme [3, 6], or an on-line key distribution center and TESLA broadcast authentication [14], or a key establishment scheme based on the mobility of the nodes and mutual node encounters [8].

We observe three parties: the claimant, the certifier and the verifier. The claimant is a node that wants to prove its encounter with some other node (the certifier) that happened before, at, or after a specific time. The certifier is either a node that certifies the time of the encounter, or a reference point that certifies the claimant's location at a given time. The verifier is a node that verifies the claimant's claim about its encounter with the certifier.

We observe the following scenario: two nodes u and v find themselves in each others' power range and want to certify their encounter. In this scenario, both nodes play the role of both the certifier and the claimant: node u is a certifier for node v and node v is a certifier for node u. There are several ways in which u and v can convince the verifier that they have indeed met. First, one of the nodes (e.g., u) sends through an authentic channel a message to the verifier that it has met v. To believe that u and v have indeed met, the verifier needs to trust u, otherwise, u can cheat. Second, both nodes u and v can send messages to the verifier stating that the encounter took place. When the verifier receives both messages, it can believe that u and v met, unless they colluded. The third approach is that during their encounter, the nodes exchange proofs of their encounter. In this case, each node can separately, and at any later time, prove to the verifier that it encountered the other node.

We follow the third approach, as it is the most appropriate for systems with frequent node encounter certifications, in which the nodes need to be able to prove their encounters individually to any other node. We note that the first two approaches might be convenient for systems in which communication between the nodes can always be guaranteed, and in which only a small fraction of encounters are verified (otherwise it would incur a large communication overhead).

We observe two verification scenarios, *any-to-any* and *any-to-one*. In the any-to-any scenario, all nodes in the network can perform the role of the claimant, the verifier and the certifier. More precisely, we enable each node to efficiently prove to any other node that it encountered some third node before, at or after a certain time. In the any-to-one scenario, all the nodes in the network can perform the role of the claimant and the certifier, but only a single node (or a subset of nodes) performs encounter verification. The any-to-any scenario corresponds to pure ad hoc networks, whereas the any-to-one scenario is more appropriate for multi-hop cellular networks where base stations naturally perform encounter verifications (e.g., for charging purposes).

3. MECHANISMS FOR ENCOUNTER VERIFICATION

In the following subsections, we describe the mechanisms of SECTOR; they differ both in the guarantees that they offer and in communication, computation and storage complexity. We describe first the mechanisms that bound node distance, then the mechanisms that guarantee encounter freshness and finally, the mechanisms that guarantee the time of node encounters.

3.1 Mutual Authentication with Distance-Bounding (MAD)

In this section, we propose a protocol for the mutual authentication of nodes with distance bounding that we call MAD. The MAD protocol enables the nodes to determine their mutual distance at the time of encounter. The MAD protocol is thus an important mechanism for secure encounter certification, as it prevents false encounter certification caused by wormhole or mafia fraud attacks [9].

The notion of distance-bounding protocols was first introduced by Brands and Chaum [4]. They proposed a technique that enables a party to determine a practical upper bound on its physical distance to another party. The main idea of the technique is simple but very powerful: it is based on the fact that light travels with a finite speed, and with current technology it is easy to measure (local) timings with nanosecond precision. The proposed distance-bounding technique essentially consists in a series of rapid bit exchanges between the parties. Each bit sent by the first party is considered to be a challenge for which the other party is required to send a one bit response immediately. By (locally) measuring the time between sending out the challenges and receiving the responses, the first party can compute an upper-bound on the distance to the other party.

Here, we slightly modify the distance-bounding protocol proposed by Brands and Chaum so that it better fits our requirements. First, our protocol allows both parties to measure the distance to the other party simultaneously. Second, we avoid the use of digital signatures. Since our distance-bounding protocol will be run frequently (each time nodes encounter), the use of standard digital signatures in each run of the protocol would result in an unacceptable overhead. Instead, we base authentication on symmetric key primitives. More precisely, we assume that each pair of parties share a symmetric key, that the nodes established before running the distance-bounding protocol between them. This key is used to generate message authentication codes (MAC) in order to prove the authenticity of the messages exchanged in the distance-bounding protocol. We will denote the MAC function controlled by the symmetric key k by mac_k.

Let u and v denote the two parties in the protocol, and let their shared key be k_{uv}. The protocol works as follows (see also Figure 1):

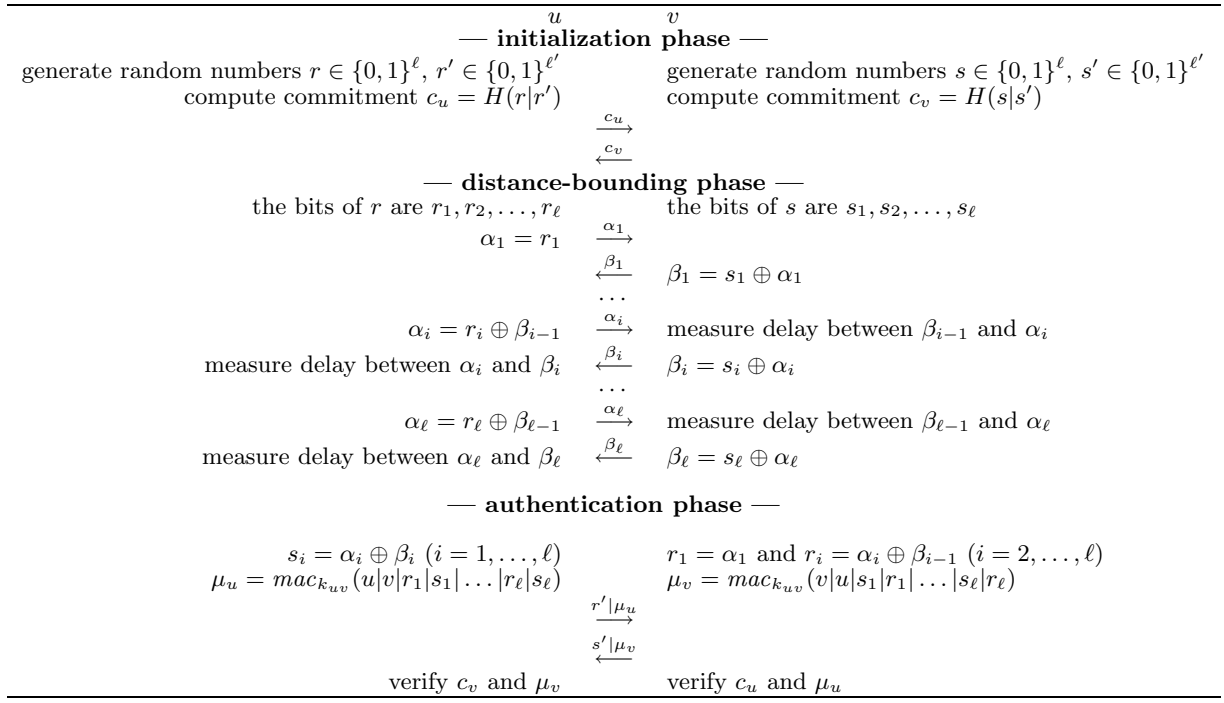

Figure 1: Operation of the Mutual Authenticated Distance Bounding protocol (MAD).

- **Initialization phase:**
  Both u and v uniformly generate two numbers at random. The numbers of u are denoted by r and r', and the numbers of v are denoted by s and s'. Both r and s are ℓ bits long, and both r' and s' are ℓ' bits long (i.e., $r, s \in \{0,1\}^{\ell}$ and $r', s' \in \{0,1\}^{\ell'}$). Both u and v compute a commitment to the generated numbers by using a collision resistant one-way hash function H: $c_u = H(r|r')$ and $c_v = H(s|s')$. Finally, u sends c_u to v and v sends c_v to u. Note that the random numbers can be generated and the commitments can be computed well before running the protocol.

- **Distance-bounding phase:**
  Let the bits of r and s be denoted by r_i and s_i $(i = 1, 2, \ldots, \ell)$, respectively. The following two steps are repeated ℓ times, for $i = 1, 2, \ldots, \ell$:

  - u sends bit α_i to v, where $\alpha_1 = r_1$ and $\alpha_i = r_i \oplus \beta_{i-1}$ for $i > 1$;
  - v sends bit $\beta_i = s_i \oplus \alpha_i$ to u immediately after it received α_i from u.

  Node u measures the times between sending α_i and receiving β_i, and v measures the times between sending β_i and receiving α_{i+1}. From the measured times, they both estimate an upper-bound on their distance.

- **Authentication phase:**
  u computes the bits $s_i = \alpha_i \oplus \beta_i$, and the MAC

  $$\mu_u = mac_{k_{uv}}(u|v|r_1|s_1|\ldots|r_\ell|s_\ell)$$

  Similarly, v computes the bits $r_1 = \alpha_1$ and $r_i = \alpha_i \oplus \beta_{i-1}$ for $i > 1$, and the MAC

  $$\mu_v = mac_{k_{uv}}(v|u|s_1|r_1|\ldots|s_\ell|r_\ell)$$

  Finally, u sends $r'|\mu_u$ to v and v sends $s'|\mu_v$ to u. u verifies that the commitment c_v and the MAC μ_v of v are correct, and v verifies that the commitment c_u and the MAC μ_u of u are correct.

In the above protocol, the MAC ensures the authenticity of the exchange: both u and v can believe that they ran the protocol with each other, and thus the distance that they estimated in the distance-bounding phase is really the distance between u and v. Sending the commitments in the initialization phase and making each bit sent in the distance-bounding phase dependent on the bit received from the other party in the previous step ensures that the parties cannot send bits too early, and thus, cannot cheat the other party by appearing to be closer than they really are.

The security of the protocol further depends on the number of bits exchanged in the distance-bounding phase. Even if functions *mac* and H are secure, an attacker can successfully compromise the protocol by guessing the value of the bits $r_1, \ldots, r_\ell$ (or $s_1, \ldots, s_\ell$), and sending the guessed bits to v (respectively u) before those bits are revealed by u (respectively v). However, the probability of a successful guess is $1/2^{\ell}$, and hence decreases exponentially in ℓ.

3.2 Guaranteeing Encounter Freshness (GEF)

Having described the SECTOR protocol for authenticated distance-bounding, we propose two mechanisms (GEF-Ce and GEF-CeCl) for guaranteeing the freshness of node encounters. By guaranteeing encounter freshness, we mean that the claimant can prove to the verifier that its encounter with the certifier happened at, or before, but not later than the time of their actual encounter.

3.2.1 GEF with Certifier Authentication (GEF-Ce)

GEF-Ce uses hash chains for guaranteeing encounter freshness. Each node u creates a hash chain $V_0, V_1, \ldots, V_N$ by choosing the initial value V_0 uniformly at random and computing $V_i = H(V_{i-1})$ for $i = 1, 2, \ldots, N$, where H is a one-way hash function. V_N is called the root of the hash chain and it is distributed to all other nodes in the network in an authentic way. Node u discloses the elements of its hash chain to its one-hop neighbors in reverse order (with respect to generation) begining with V_{N-1} and proceeding towards V_0. A simple disclosure scheme consists in publishing chain elements at regular time intervals. Thus, the length of the chain N is chosen as the expected number of hash values that will need to be disclosed. This disclosure scheme enables each node that resides in the first-hop neighborhood of u to get the latest published value V_i. A neighbor receiving V_i can verify its authenticity by hashing it iteratively $N - i$ times and comparing the result $H^{(N-i)}(V_i)$ to the pre-distributed authentic root V_N. The knowledge of V_i then serves as a proof of being close to u prior to the time when V_i was published. In other words, a verifier can typically check which of the claimants had met a given certifier more recently.

Due to its simplicity, this mechanism can be very efficiently implemented. As shown by Coppersmith and Jakobsson in [7], a single hash chain of length N can be managed by storing only $\lceil \log_2 N \rceil + \lceil \log_2(\log_2 N + 1) \rceil$ hash values and outputting chain elements at a cost of only $\frac{1}{2} \log_2 N$ hash operations per element. The mechanism also requires that each node stores $n - 1$ hash chain roots (the authentic hash chain roots of the other nodes). The cost of verification of a received hash value is at most N hash function computations, but this can be significantly reduced if, instead of storing hash chain roots, each node stores the most recently received authentic hash chain elements of the other nodes.

Clearly, GEF-Ce mechanism provides only certifier, but not claimant authentication (this means that it can be verified who disclosed a given hash value, but it cannot be verified to whom the value was disclosed).

3.2.2 GEF with Certifier and Claimant Authentication (GEF-CeCl)

GEF with Certifier Authentication, although appealing due to its simplicity, lacks claimant authentication and thus provides only weak protection against malicious nodes (we will discuss this in Section 4.2). With multiple hash-chains, we will now enable higher protection against attacks, but at a somewhat higher storage cost. The mechanism works as follows. Each node u creates $n - 1$ different hash chains, one chain for each of the other $n - 1$ nodes in the network. The hash chain that node u associates to node v is created in a way that u chooses an initial value V_0^v, and computes its corresponding root value V_N^v. While u keeps the initial values secret, it distributes all the root values of the chains to the other nodes, along with the corresponding addresses of the nodes.

To certify their encounter, two nodes first exchange their addresses and then the values of the hash chains that they previously created for each other and for that time instance. In this way, by receiving a hash value generated by the other node, any of the two nodes can prove that it has indeed encountered the other node prior to that time. The main advantage of this mechanism over the single hash chain mechanism is that it enables claimant authentication during the verification. Another advantage is that its computation complexity does not change with the number of nodes and is the same as for the GEF-Ce mechanism. Its main disadvantage is the storage requirement, which increases linearly with the number of nodes in the network and is thus n times higher than that for the GEF-Ce mechanism: $O(n \log_2 N)$.

Both GEF-Ce and GEF-CeCl mechanisms are designed for any-to-any verification, but can be applied equally to any-to-one scenarios, in which case the cost of distributing hash chain roots is significantly reduced.

3.3 Guaranteeing the Time of Encounter (GTE)

Having presented two simple mechanisms based on hash chains, we now propose more sophisticated mechanisms based on hash-trees that provide guarantees on the exact time of the encounters.

The mechanism of tree-authenticated values is an efficient hash tree authentication mechanism. It was first presented by Merkle and it is also known as Merkle hash trees [19]. To authenticate values $(V_0, V_1, ..., V_N)$, we place them at leaf nodes of a binary tree. We then use the Merkle hash tree construction to commit to the values $V_0, ..., V_N$, which works in the following way. First, each value is hashed to avoid disclosing the neighbor values during authentication. Thus, each value V_i hashes into $m_i = H(V_i)$. Each internal node of the binary tree is derived from its two child nodes. Consider the derivation of some parent node m_p from its left and right child nodes m_l and m_r: $m_p = H(m_l|m_r)$. We compute the levels of the tree recursively from the leaf nodes to the root node.

3.3.1 GTE with certifier authentication (GTE-Ce)

We use the Merkle trees to guarantee the time of encounter. Our scheme works as follows. Each node concatenates value release times with random values to create the leaf values $V_0, V_1, ..., V_N$. Thus, we write $V_i = time_i|rand_i$, where $time_i$ represents the time at which the value V_i will be released and $rand_i$ is a random value generated by the node for the i'th leaf value of the tree. After these initial values are generated, they are blinded with a one-way hash function to prevent disclosing neighboring values in the authentication information. Thus, $m_i = H(V_i) = H(time_i|rand_i)$. Once it generates the tree, each node distributes the root of the tree to other nodes in an authentic way. The node then releases the time values in the order from V_0 to V_N (e.g., in fixed time intervals, starting at a predefined time), along with their siblings on the tree; these values enable any other node to recompute the root of the tree and thus to authenticate the values.

Example (Figure 2): When a certifier releases the value V_2, it releases it along with its siblings m_3, m_{01} and m_{47}. These values are then stored by the claimants in the certifier's neighborhood. When a claimant presents the value

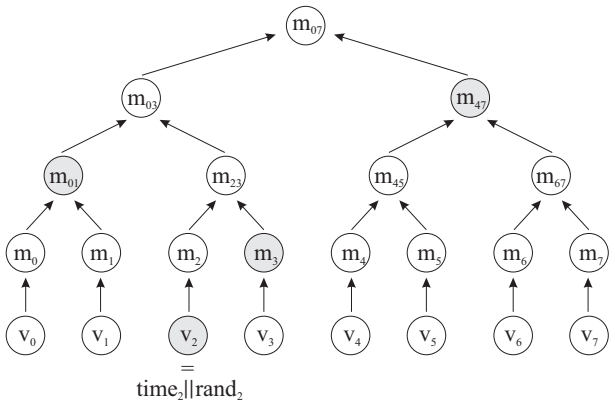

Figure 2: Example of the Merkle hash tree and the authentication of V_2 with the GTE-Ce mechanism.

V_2 to a verifier as a proof that it encountered the certifier at time t, the verifier authenticates the received value and extract from it the time of the encounter ($time_2$ in the example). To authenticate the received value V_2, the verifier computes $H(H(m_{01}|H(H(V_2)|m_3))|m_{47})$ and checks if it corresponds to the previously received authentic root of the certifier m_{07}. Similarly to GEF-Ce, this mechanism can also be implemented very efficiently. As shown in [18], Merkle trees can be efficiently stored by storing less then $1.5 \log_2^2 N / \log_2 \log_2 N$ hash values, and hash tree values can be efficiently outputted with a computation cost of less then $2 \log_2 N / \log_2 \log_2 N$ hash function operations per output value. The mechanism also requires that each node stores $n-1$ root hash values.

3.3.2 GTE with certifier and claimant authentication (GTE-CeCl)

The same security problems as with GEF-Ce are inherent to GTE-Ce, because in both mechanisms only the certifier is authenticated. The main problem is that with GTE-Ce it is only possible to verify the source of the message (the certifier), but the verifier cannot be sure that the encounter really happened with the claimant or with some other node that disclosed the authentic value to the claimant. In order to prevent this and similar attacks, we can use a similar mechanism as with hash chains. Each node creates $n-1$ hash trees, one tree for each other node in the network. Consequently, instead of distributing one, a node is suppose to distribute $n-1$ root values, one for each node, by thus enabling authentication during certification and verification of the encounter. We call this mechanism GTE-CeCl-basic. However, creating $n-1$ trees and distributing the same number of roots is expensive, as it requires storing $n \times 1.5 \log_2^2 N / \log_2 \log_2 N$ tree values and $(n-1)^2$ roots per each node.

Certifier and claimant authentication can be achieved more efficiently, whereas each node still maintains a single hash-tree. We propose that instead of creating $n-1$ hash trees of size N (N leaf nodes), each node creates a single tree of size $n \times N$ and divides it into n equal parts. Thus, upon encounters, the certifier releases to the claimant values that are allocated for this claimant (e.g. for node i, the values in range $[(i-1)N, iN\rangle$). We call this mechanism the optimized GTE-CeCl and we denote it by GTE-CeCl-opt. The operation of GTE-CeCl-opt is shown on Figure 3. GTE-CeCl-opt is very efficient in terms of storage, as it requires each node to store only $1.5 \log_2^2 nN / \log_2 \log_2 nN$ hash values and $n-1$ hash tree roots, versus $n1.5 \log_2^2 N / \log_2 \log_2 N$ hash values and $(n-1)^2$ roots for GTE-CeCl-basic. The computational cost per output value of GTE-CeCl-basic is the same as for the GEF-Ce mechanism (less than $2 \log_2 N / \log_2 \log_2 N$ hash operations), whereas in the case of GTE-CeCl-opt, the computational cost per output value is slightly higher ($2 \log_2 nN / \log_2 \log_2 nN$ hash operations).

An example of the GTE-CeCl mechanism is shown on Figure 4. In the figure, the values V_{iv}^u and m_{iv}^u represent the values on the hash tree created by node u, which are used by u to certify its encounters with node v at time instance i. In the example that is shown on the figure, we assume that each node allocates only 8 leaf (time) values for each other node in the network. This corresponds to the example trees shown on Figures 2 and 3. In the protocol, the nodes release values (V_{2u}^v, V_{2v}^u) associated with the time $t = 2$s, given that the values are released each second.

GTE-Ce, GTE-CeCl and GTE-CeCl-opt are designed primarily for any-to-any verification scenarios and are thus very appropriate for mobile ad hoc networks.

3.3.3 Conventional symmetric-key and public-key mechanisms

A more conventional approach to solving the addressed problems would rely on classical symmetric-key and public-key techniques. As we will show, both have some drawbacks with respect to the solutions that we described: public-key cryptography because of its high computation cost, and symmetric-key cryptography because it can hardly be implemented for any-to-any verification scenario.

If public-key cryptography is used, the certifier certifies the encounter by computing a signature over a timestamp and by distributing it to its neighbors. An authenticated version of this mechanism also includes the authentication of each neighbor and the computation of a signature over the timestamp and identity of each neighbor, such that each neighbor receives a distinct message. The verification of the encounter in this mechanism is straightforward if the verifier knows the authentic public key of the certifier. A more secure mechanism assumes that nodes jointly sign the timestamp and their respective identities, when they encounter, which prevents cheating, unless the nodes collude. The public-key implementation is equally appropriate for both any-to-any and any-to-one encounter verification. Its main disadvantage is the high cost of public-key operations, which are almost three orders of magnitude slower than conventional symmetric-key and hash-chain operations.

A similar approach can be based on symmetric-key cryptography. However, a symmetric-key approach is more appropriate for any-to-one than for any-to-any verification and thus we describe its operation in any-to-one scenario.

The certifier computes a message authentication code over

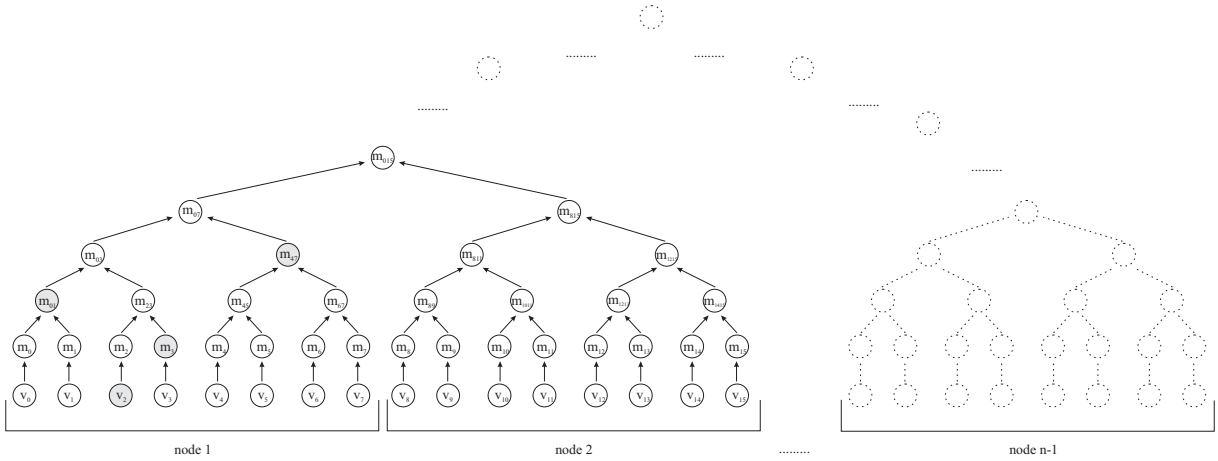

Figure 3: Tree authenticated values with the optimized GTE-CeCl mechanism. Each node creates a single hash tree with $n \times N$ leaf values: N values for each node in the network. Upon encounter, the certifier releases a value (and its siblings) that corresponds to the claimant that it encountered and to the time instance at which they encountered.

a timestamp, with a key that it shares with a base station and sends it along with the timestamp to its neighbors. The certifier's neighbors store these values and use them to prove their encounters to the base station. The authenticated version of this mechanism assumes mutual authentication of the encountering nodes by means of their shared secret key (e.g., by using MAD). After they authenticate each other, the nodes compute MACs over the timestamp and their id's with the keys that they share with the base station. These MACs then serve as proofs of the encounter. The advantage of this symmetric key based mechanism resides in its simplicity and its resilience to various attacks. Its main drawback is that it is mainly limited to any-to-one verification scenarios.

3.4 Distribution of the authentic roots

The mechanisms described in the previous subsections require each node to distribute its hash chain or hash tree roots in an authentic way to every other node in the network. Here, we briefly describe how this can be performed under the assumption that there is an off-line central authority in the system. We explain the mechanisms in terms of distributing the root of a single hash chain of each node, but they can also be used to distribute multiple hash chain roots and single or multiple hash tree roots in an identical way. In all the approaches presented below, each node maintains two hash chains: an *active* and a *pending* one. We assume that the root of the active hash chain has already been distributed; thus the active hash chain can be used for guaranteeing encounter freshness as described in the previous subsections. In contrast, the root of the pending hash chain has not been distributed yet, and the goal of the node is to distribute it to every other node in the network before the active hash chain runs out of elements. When the root of the pending hash chain has been distributed, the node can turn the pending hash chain into an active state. At the same time, the node would generate a new pending hash chain and begin distributing its root. Putting in place the pending hash chains while using the active ones ensures continuous operation of the system. We now describe three possible approaches.

In the first approach, we assume that the off-line central authority issues public-key certificates to every node in the network at initialization time. In order to distribute the root of its pending hash chain, node u digitally signs the root and floods the network with the signed message and its public-key certificate. Each node will receive the new hash chain root of u and authenticate it by verifying the signature using the public-key certificate of u. We envisage using this approach when the hash chains are long (e.g., they can be used for days without running out of elements); as a result, signature verifications must be performed only rarely.

The second approach is similar to the first, but it can also be used when the hash chains are short (e.g., they can only be used for several hours). In this approach, too, node u floods the network with the signed root of its pending hash chain, but in this case, TESLA [22] is used for signing. More specifically, u signs the root $V^{(i)}$ of its i^{th} pending hash chain with its i^{th} TESLA key $K^{(i)}$, and floods the signed message with its previous TESLA key $K^{(i-1)}$ attached to it. With the disclosed TESLA key $K^{(i-1)}$, every node can authenticate the root $V^{(i-1)}$ of the $(i-1)^{th}$ pending hash chain sent by u in the previous flood, which would then become active. Since TESLA is based on symmetric key cryptographic primitives, authentication of the hash chain roots can be done efficiently. The roots of the TESLA key chains can be distributed by messages that are signed using public-key cryptography, because TESLA key chains can last for a longer period of time, roots of TESLA key chains can be distributed less frequently.

In the third approach, the root of the pending hash chain is disseminated in an authentic way by the mobility-based scheme we proposed in [8]. Together with the root of the pending hash chain, the nodes also disseminate a time t in the future. The value of t should be estimated in such a way that the active hash chain does not run out of elements by t

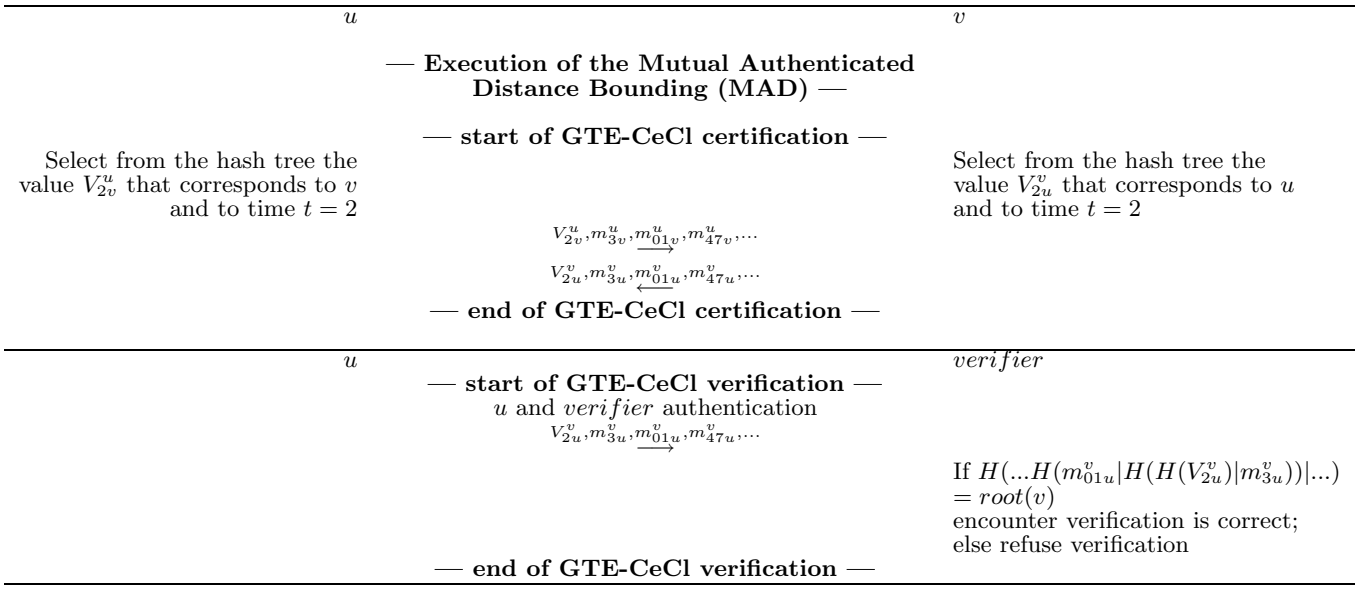

Figure 4: An example of the execution of the protocol for guaranteeing the time of node encounters (GTE-CeCl).

and the root of the pending hash chain is distributed to all other nodes by time t. Then, at t, the pending hash chain becomes active, and a new pending hash chain is generated; the process is then repeated.

4. SECURITY ANALYSIS

Having presented the protocols, we now analyze their resistance to various attacks.

4.1 Attacker model

We call a node *malicious* if it is not controlled by the central authority, but is controlled by an attacker (and thus cannot positively authenticate itself to honest network nodes). We call a node *compromised* if it can positively authenticate itself to honest network nodes, but is controlled by an attacker. We assume that when a node is compromised, its secret keys and the other secrets that it shares with other nodes become known to the attacker. Thus, a compromised node is, for other nodes, indistinguishable from an honest node. We further assume that when a node is compromised, this is not detected by other honest nodes, nor by the central authority (at least for some time).

We distinguish attackers according to the number of malicious and compromised nodes that they control. By Attacker-x-y we denote the attacker that controls x malicious and y compromised nodes [14].
We focus on four types of attacks:

- **Attack-Cl:** The attacker plays the role of a claimant and tries to convince an honest verifier that it (the attacker) has encountered an honest certifier at some time, whereas it really did not, or it did but at some different time.
- **Attack-Ce:** The attacker plays the role of a certifier and tries to convince an honest claimant that it (the attacker) is an honest certifier, or/and tries to falsify the time of the encounter.
- **Attack-CeCl:** The attacker controls two nodes: it plays a role of a claimant and the role of a certifier and tries to convince an honest verifier that the certifier and the claimant met, whereas they really did not, or if they have indeed met, the attacker tries to convince the verifier that they have met at some different time from the time of the actual encounter.
- **Attack-V:** The attacker plays the role of the verifier and tries to extract some encounter information from an honest claimant in order to use this information to prove to an honest verifier that it has encountered those nodes that the honest claimant met.

Some of the proposed mechanisms contain the mutual or one-way authentication of nodes during certification and/or verification. For all mechanisms, we implicitly assume that the verifier and the claimant always perform mutual authentication before they verify/prove the encounter. The claimant and certifier authentication during certification of the encounter is not implemented in all mechanisms. Mechanisms such as GEF-Ce, GEF-CeCl, GTE-Ce and GTE-CeCl implicitly contain the authentication of the certifier by the claimant, but not the authentication of the claimant by the certifier. Only the MAD mechanism assumes mutual authentication between the claimant and the certifier. So, only a combination of MAD with time and freshness mechanisms will ensure the full security of the system in which they are implemented.

4.2 Resistance to attacks

4.2.1 GEF-Ce and GTE-Ce

GEF-Ce and GTE-Ce include only the authentication of the certifier by the claimant, but not the authentication of the

claimant by the certifier. Thus, any node can easily get authentic values from the certifier, and then distribute it to other nodes. This allows the attacker to share the receiver hash value among all the nodes that it controls, which they can use to successfully prove that they have met the certifier at that time (Attack-Cl). Moreover, an attacker that controls a single compromised node can successfully attack GEF-Ce and GTE-Ce mechanisms by requesting some node to prove its encounters, and by later using these values as a proof that it itself encountered these nodes (Attack-V). Finally, an attacker can successfully perform Attack-Ce against GEF-Ce and GTE-Ce and can convince an honest claimant and an honest certifier that they have met, even if they have never have been in each other's power range. This is achieved by creating a wormhole between two honest parties (Attack-Ce).

GEF-Ce and GTE-Ce mechanisms are thus vulnerable to all attacks performed by attackers that control several malicious nodes (Attacker-x-0), and even to attacks performed by attackers that control a single compromised node (Attacker-0-1). These mechanisms are only resistant to attacks from the attackers that control a single malicious (but not compromised) node (Attacker-1-0).

4.2.2 GEF-Ce and GTE-Ce with MAD

GEF-Ce and GTE-Ce mechanisms with authenticated distance bounding (MAD) are more resistant to attacks, because MAD provides both distance bounding and mutual authentication between the certifier and the claimant. With MAD, GEF-Ce and GTE-Ce are resistant to Attack-Cl and Attack-Ce attacks performed by an attacker that controls multiple malicious nodes (Attacker-x-0) and attacks performed by an attacker that controls a single compromised node (Attacker-0-1). GEF-Ce and GTE-Ce with MAD are also resistant to Attack-V, performed by an attacker that controls multiple malicious nodes (Attacker-x-0), but not to attacks from Attacker-0-1.

4.2.3 GEF-CeCl and GTE-CeCl with MAD

We analyze GEF-CeCl and GTE-CeCl mechanisms with the MAD mechanism. These two mechanisms are stronger then GEF-Ce and GTE-Ce in that they bind the released hash values to the identity of the claimant, so that this value cannot be reused by any node but the claimant for proving an encounter with the certifier. More precisely, the hash value released by the certifier uniquely binds the certifier and the claimant. Due to this improvement, GEF-CeCl and GTE-CeCl with MAD are resistant to Attack-Cl and Attack-V attacks performed by Attacker-x-y. By this we mean that it does not matter how many malicious or compromised nodes the attacker controls: it cannot convince an honest verifier to believe a malicious or compromised claimant, if the certifier is honest. In the same way, a malicious verifier cannot extract any information from an honest claimant that can help him to prove anything except that the claimant met an honest certifier; this is guaranteed by the hash values that uniquely bind the certifier, the claimant and the time of the encounter. GEF-CeCl and GTE-CeCl with MAD mechanisms are also resistent to Attack-Ce attacks performed by an Attacker-0-1, and by an Attacker-x-y.

If a single certifier is compromised, it cannot convince an honest claimant that they have met, unless they have indeed met. The GEF-CeCl and GTE-CeCl with MAD are resistent to Attack-Ce and Attack-Cl performed by Attacker-x-y in a broader sense, as the only way to cheat for the attacker is to delegate the same identity to several of the nodes that it controls. However, one of its nodes still needs to meet the honest claimant (or certifier) to be able to claim that all of its nodes have met the certifier (or the claimant), but they claim this only under the same identity used for the encounter. Moreover, multiplying or exchanging identities can be detected by the nodes through a consistency check of the encounters.

4.2.4 Public-key and Symmetric-key mechanisms

These mechanisms typically exhibit the same level of security as GTE-CeCl and GEF-CeCl mechanisms with MAD, as they provide the same authentication between the certifier, verifier and the claimant.

4.2.5 Other attacks

Other attacks can be envisioned against the proposed mechanisms, especially Attack-CeCl, which has not been discussed so far. In this attack, an attacker controls two compromised nodes and can easily convince any verifier that these two nodes have met, even if they did not. However, through topology tracking and consistency checking, these false encounters can be detected, especially if the nodes actively participate in the network operation. There is always a chance that an attacker can remove its nodes from the network range and claim that they mutually encountered. At the same time, its nodes did not encounter other network nodes. We do not know how useful this attack can be for the attacker. As we already discussed, additional attacks can be performed by an attacker that uses the same (compromised) identity at several nodes. This attack also cannot be prevented, but only detected through topology tracking. However, Attack-Ce by Attacker-x-1 is not very powerful as it is limited to a single attacker, which can be isolated through some reputation mechanisms [5].

5. PERFORMANCE ANALYSIS

Here we analyze in more detail the storage, the computation and the communication overhead of the proposed mechanisms; the summary is shown on Figure 5. From this figure we can conclude that the mechanism GEF-Ce has the lowest cost, but also provides the lowest level of protection against attacks and provides only freshness guarantees, whereas the GTE-CeCl mechanism provides the highest level of protection and the exact time guarantees, but at a slightly higher cost.

We consider a mobile ad hoc network of $n = 100$ nodes, where the hash chain roots are updated approximately every 1.5 days (36h) and hash values are released every second. In this network, GEF-Ce mechanism induces the following costs. Each node stores $17+99$ hash values ($\log_2$ of the number of seconds in 1.5 days (i.e., $N = 1.5 \times 24 \times 3600$) $+ n - 1$ (99) root values), and 99 secret keys; this means that each node stores less then 4kB of information. The computation cost is equally small, since to release a hash value, each node needs to perform only 8 hash operations ($\frac{1}{2}\log_2$ of the number of seconds in 1.5 days, i.e., the number of values on the

Mechanism	Storage cost	Computation cost	Communication cost
GEF-Ce	$\log_2 N + (n-1)$	$\frac{1}{2}\log_2 N$	$O(1)$
GEF-CeCl	$n \log_2 N + (n-1)^2$	$\frac{1}{2}\log_2 N$	$O(1)$
GTE-Ce	$1.5 \log_2^2 N / \log_2 \log_2 N + (n-1)$	$2 \log_2 N / \log_2 \log_2 N$	$O(\log_2 N - 1)$
GTE-CeCl-basic	$n1.5 \log_2^2 N / \log_2 \log_2 N + (n-1)^2$	$2 \log_2 N / \log_2 \log_2 N$	$O(\log_2 N - 1)$
GTE-CeCl-opt	$1.5 \log_2^2 nN / \log_2 \log_2 nN + (n-1)$	$2 \log_2 nN / \log_2 \log_2 nN$	$O(\log_2 nN - 1)$
public-key	$O(n)$	1 signature	$O(1)$
symmetric-key (any-to-one)	$O(n+1)$	1 encryption	$O(1)$

Figure 5: Communication cost per certification per node, computation cost per certification per node, and storage cost per node for proposed mechanisms (n: number of nodes; N: number of time intervals for which the hash chain or Merkle tree is pre-computed).

chain). This computation cost is very small since already 400 MHz Pentium II processors running Windows can perform more then 125,000 hash operations per second [21]. It is also important to mention that 400 MHz processors are already available today in PDAs (e.g., HP pocket PCs). The communication cost of GEF-Ce is also low, as this mechanism assumes the exchange of a single hash value (e.g., 160 bits) per encounter. It is worth noticing that the GEF-Ce, as well as the GTE-Ce mechanism cost, scales well with the size of the network, given that the communication and computation costs of these mechanisms do not depend on the number of nodes in the network.

If a higher security and time guarantee is needed and the GTE-CeCl-opt mechanism is implemented, the network costs are somewhat higher. The storage cost per node is then approximately $110 + 99$ hash values and 99 secret keys (less then 6kB) (Figure 5). The corresponding computation cost per hash tree value is around 9 hash chain operations. The communication cost of the GTE-CeCl-opt is higher than with GEF-Ce or GTE-Ce and it amounts to 16 hash values, or 320 bytes per encounter per node. This cost can be significantly reduced at the expense of initially distributing to the nodes not only hash tree roots, but also several lower hash tree layer values. This approach somewhat increases the storage cost per node, but reduces the cost of all subsequent communication.

Although the symmetric-key mechanism has the lowest communication and storage cost it is only suitable for any-to-one encounter verifications, typically for multi-hop cellular networks. The public-key based mechanism also exhibits very low communication and storage costs, but is very inefficient in terms of computation cost, as the public-key operations are approximatively three orders of magnitude slower than the symmetric-key and hash operations [21].

One additional figure of merit of the proposed mechanisms is also their cumulative storage cost, meaning the cost of storing encounter proofs received by other nodes. For GEF-Ce and GTE-Ce, this cost is exactly the number of encounters multiplied by the hash value size (160 bits). This cost cannot be precisely measured because it depends on the rate of change of the network topology (notably on the number of different nodes that a node encounters in a given period, and on the desired granularity of the encounter tracking).

We illustrate this cost with a simple example. We assume that the granularity of the encounter tracking is 1 second, which means that the nodes update the encounters with their neighbors every 1 second. This would mean that in 1.5 days (36 h), each node stores 2.5MB of hash chain values. However, it is not necessary that a node stores an encounter with another node every second (i.e., the corresponding hash values), especially if two nodes stay in each others' power range for longer period of time. Instead, a node can store only a fraction of the encounters with each node, if these encounters are consecutive (e.g., for a whole minute). By storing only a fraction of the encounters with other nodes, nodes can significantly reduce their cumulative storage cost (e.g., if a node stores only 2 encounters with the same node per minute, it will need to store less than 300kB instead of 2.5MB). Even if a very fine granularity of encounters is required and a node saves the proofs of every encounter and the storage required is 2.5MB, for a whole mechanisms can still be implemented, as today's personal portable devices are already equipped with more than 128MB of storage space. The other aspect of the cumulative storage cost is the question of the necessity to keep 1.5 days of old network topology information. This and related issues will be a part of our future work.

6. APPLICATIONS

In this section we briefly present several examples of the applications of SECTOR.

6.1 Prevention of wormhole attacks

Wormhole attacks in wireless networks were recently discussed by several researchers, including Dahill et al. [24], Papadimitratos and Haas [20], and Hu, Perrig and Johnson [12]. A wormhole is a fast tunnel (e.g., a wireline link) between two nodes that are, typically, physically very far from each other. Without this link, it would take several hops for a packet to be transmitted between the nodes, whereas through a wormhole, the transmission is very fast and it requires only one hop. Wormholes are normally very useful as they enable faster communication between nodes, but they can be used by malicious users to prevent the correct operation of routing protocols. In [12], the authors propose a mechanism called "packet leashes" that aims at preventing wormhole attacks by making use of the geographic location of the nodes (geographic leashes), or of the transmission time of the packet between the nodes (temporal

leashes). In the latter mechanism, the authors assume that the internal clocks of the nodes are precisely synchronized.

In our approach, we make no assumptions about clock synchronization between nodes, nor do we assume that the nodes are equipped with any positioning devices. To detect wormhole attacks, we use our MAD protocol. This protocol applies the same principle as packet leashes, with the difference that it measures the distance at a single node, unlike with packet leashes where the distance is measured by calculating the difference in time or location at both nodes. MAD has another important advantage over packet leashes: that each node can perform distance bounding without having to trust an other party, which is not the case in packet leashes, where two nodes detecting wormholes have to trust the exchanged information (time or location). Another way our mechanisms can help to detect wormholes in wireless networks is through topology and encounter tracking with GTE mechanisms. If a base station or a node collects network topology information, it can also identify wormhole links by comparing the obtained encounter information.

6.2 Topology tracking

The first application that we consider is topology tracking. Topology tracking can be performed by the base station (in multi-hop cellular) or by the nodes themselves (in pure ad hoc networks).

6.2.1 Multi-hop Cellular Networks

In the case where a base station performs topology tracking, we use the MAD protocol for distance bounding between the nodes, and symmetric-key mechanisms to certify the time of the encounter. When nodes u and v receive each others' hello messages, they first run the MAD protocol to verify if their mutual distance is smaller then their power range. If this is the case, u computes a MAC $MAC_{K_{uBS}}(u, v, timestamp_u, d_{uv})$, where K_{uBS} is the shared key between u and the base station (BS) and d_{uv} is the distance between u and v estimated based on the time-of-flight measurements performed by u. Node v similarly computes $MAC_{K_{vBS}}(v, u, timestamp_v, d_{vu})$. The nodes then exchange the computed MACs and the values that they contain. These proofs of encounters can be either passed to the base stations when the nodes get in their power range, or be periodically sent to the base stations by the nodes. Furthermore, the base stations can periodically pool the nodes for the latest encounters.

From the collected encounter information, the network authority can then reconstruct the history of the node encounters and thus the history of the network topology. This information can be very useful for the network operator, to observe the node behavior, to prevent and to detect security breaches, or to identify cheating nodes [2, 17].

6.2.2 Pure ad Hoc Networks

To enable topology tracking in pure ad hoc networks, we use mechanisms different than in the multi-hop cellular networks, notably, we use GTE-CeCl with MAD. Like in multi-hop cellular networks, the MAD mechanism ensures distance bounding between nodes. But unlike in hybrid networks, here we cannot use the symmetric-key mechanisms as they do not allow an efficient any-to-any verification of the encounters. Instead, we use GTE-CeCl.

Every node creates a single Merkle tree of $(n-1) \times N$ leaf values, N for each node in the network. The root of the tree is then distributed to the other nodes, along with the first and the last leaf value that is allocated for each node (see Figure 3). When two nodes u and v meet, they first run the MAD distance bounding protocol and then exchange hash values, u discloses the value that corresponds to time t and to node v, along with its siblings on the tree. These values are then stored by v and serve as a proof to any other node in the network that u and v have indeed met. v performs a similar operation and discloses the tree value, which corresponds to time t and node u. This value serves to u as a proof that it has indeed met v. Any node in the network can then collect the proofs of encounters, verify them and use them to detect node misbehavior or cheating.

6.3 Security of Last Encounter Routing

In [10], Dubois-Ferriere, Grossglauser and Vetterli present an approach called FRESH, for efficient route discovery in mobile ad hoc networks using encounter ages. They show that if nodes only keep track of the time of their encounters, route discovery can be performed at a much lower cost than with traditional broadcast search methods. Each node maintains a local database of the time of its last encounter with other nodes in the network. This database is consulted by packets to obtain estimates of the destination's last encounter. As a packet travels towards its destination, it is able to successively refine this estimate, because node mobility has "diffused" estimates of the times of encounters. In each step of the destination search, the node that receives the packet performs a restricted broadcast and queries its neighbor nodes for the last encounter with the destination.

In the original proposal, this route discovery mechanism is not secured and a dishonest node can for example advertise a very recent encounter with the destination of a given packet and thus prevent the packet from ever reaching the destination. To prevent this, we apply our GEF-Ce or GEF-CeCl mechanisms with MAD. Whenever a node broadcasts a request for the node that had the most recent encounter with the destination, all nodes need to reply with the hash value that proves this encounter. The node with the most recent correctly verified encounter is then chosen as a relay.

Introducing GEF-Ce or GEF-CeCl into route discovery is not very costly because it requires, besides node authentication through MAD, the exchange of only a few hash values.

7. CONCLUSION

In this paper, we have presented SECTOR, a set of protocols for the secure verification of the time of encounters between nodes. We have built these protocols on well-established cryptographic techniques, including hash chains and Merkle hash trees. We have also shown how to adapt the protocols to the specific requirements of a given application. We have explained that the overhead is very reasonable and we have assessed the robustness with respect to attackers of different degrees of strength. We have applied this solution to several problems, including prevention of wormhole attacks,

securing routing protocols based on last encounters, as well as cheating detection by means of topology tracking.

To the best of our knowledge, this paper is the first to address the problem of securing topology and encounter tracking; the only exception is the prevention of the wormhole attack, which was previously investigated by other researchers.

In terms of future work, we intend to study in more detail the behavior of the proposed protocols, notably by means of simulations, in different mobility scenarios. We will also show that this approach can be useful in more conventional, one-hop wireless networks, if the base stations (or the access points) are not completely trusted.

8. REFERENCES

[1] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in ad hoc wireless networks. In *Proceedings of NDSS*, 2002.

[2] N. Ben Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of MobiHoc*, 2003.

[3] R.B. Bobba, L. Eschenauer, V.D. Gligor, and W. Arbaugh. Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks. Technical Report TR 2002-44, University of Maryland, May 2002.

[4] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *Theory and Application of Cryptographic Techniques*, pages 344–359, 1993.

[5] S. Buchegger and J. Y. Le Boudec. Performance analysis of the confidant protocol (cooperation of nodes - fairness in dynamic ad-hoc networks). In *Proceedings of MobiHoc 2002*, Lausanne, June 2002.

[6] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[7] D. Coppersmith and M. Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of the Fifth Conference on Financial Cryptography (FC '02)*, 2002.

[8] S. Čapkun, J.-P. Hubaux, and L. Buttyán. Mobility Helps Security in Ad Hoc Networks. In *Proceedings of MobiHoc*, 2003.

[9] Y. Desmedt. Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In *SecuriCom'88*, 1988.

[10] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages. In *Proceedings of MobiHoc*, 2003.

[11] R. Hauser, A. Przygienda, and G. Tsudik. Reducing the Cost of Security in Link State Routing. In *Proceedings of NDSS*, February 1997.

[12] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *Proceedings of IEEE Infocom*, April 2003.

[13] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, June 2002.

[14] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Proceedings of MobiCom*, September 2002.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson. Efficient Security Mechanisms for Routing Protocols. In *Proceedings of NDSS*, February 2003.

[16] J.-P. Hubaux, Th. Gross, J.-Y. Le Boudec, and M. Vetterli. Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project. *IEEE Communications Magazine*, January 2001.

[17] M. Jakobsson, J.-P. Hubaux, and L. Buttyán. A Micropayment Scheme Encouraging Collaboration in Multi-hop Cellular Networks. In *Proceedings of the 7th Financial Cryptography Conference*, 2003.

[18] M. Jakobsson, T. Leighton, S. Micali, and M. Szydlo. Fractal Merkle Tree Representation and Traversal. In *RSA Cryptographers Track*, 2003.

[19] R. C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1980.

[20] P. Papadimitratos and Z.J. Haas. Secure Routing for Mobile Ad Hoc Networks. In *Proceedings of CNDS*, January 2002.

[21] M. Peirce. *Multi-Party Electronic Payments for Mobile Communications*. PhD thesis, 2000.

[22] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, 5(Summer), 2002.

[23] K. Romer. Time Synchronization in Ad Hoc Networks. In *Proceedings of MobiHoc*, 2001.

[24] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A Secure Routing Protocol for Ad hoc Networks. In *Proceedings of ICNP*, 2002.

[25] N. Sastry, U. Shankar, and D. Wagner. Secure Verification of Location Claims. Technical Report UCB//CSD-03-1245, EECS, UCB, 2003.

[26] F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, February 2002.

[27] B. Waters and E. Felten. Proving the Location of Tamper-Resistant Devices. Technical report, Princeton University, http://www.cs.princeton.edu /∼bwaters/research/location_proving.ps

[28] K. Zhang. Efficient Protocols for Signing Routing Messages. In *Proceedings of NDSS*, March 1998.