Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2453–2467
computer communications
www.elsevier.com/locate/comcom

Toward secure and scalable time synchronization in ad hoc networks

Lin Chen *, Jean Leneutre

Département INFRES – CNRS LTCR-UMR 5141, École Nationale Supérieure des Télécommunications, 37–39, Rue Dareau, 75014 Paris, France

Available online 5 May 2007

Abstract

Time synchronization is crucial in ad hoc networks. Due to the infrastructure-less and dynamic nature, time synchronization in such environments is vulnerable to various attacks. Moreover, time synchronization protocols such as IEEE 802.11 TSF (Timing Synchronization Function) often suffer from scalability problem.

In this paper, we address the security and the scalability problems of time synchronization protocols in ad hoc networks. We propose a novel suite of time synchronization mechanisms for ad hoc networks based on symmetric cryptography. For single-hop ad hoc networks, we propose SSTSP, a scalable and secure time synchronization procedure based on one-way Hash chain, a lightweight mechanism to ensure the authenticity and the integrity of synchronization beacons. The "clock drift check" is proposed to counter replay/delay attacks. We then extend our efforts to the multi-hop case. We propose MSTSP, a secure and scalable time synchronization mechanism based on SSTSP for multi-hop ad hoc networks. In MSTSP, the multi-hop network is automatically divided into single-hop clusters. The secure intra-cluster synchronization is achieved by SSTSP and the secure inter-cluster synchronization is achieved by exchanging synchronization beacons among cluster reference nodes via bridge nodes.

The proposed synchronization mechanisms are fully distributed without a global synchronization leader. We further perform analytical studies and simulations on the proposed approaches. The results show that SSTSP can synchronize single-hop networks with the maximum synchronization error under 20 μs and MSTSP 55–85 μs for multi-hop networks, which are, to the best of our knowledge, among the best results of currently proposed solutions for single-hop and multi-hop ad hoc networks. Meanwhile, our approaches can maintain the network synchronized even in hostile environments.

Keywords: Time synchronization; Ad hoc networks; Security; Scalability

1. Introduction

Ad hoc networks are autonomous collections of mobile nodes communicating with each other over wireless links and cooperating in a distributed manner in order to provide the necessary network functionality in the absence of a fixed infrastructure. In such environments, time synchronization is crucial. It is a key function to perform power management and to support the medium access control protocol in the Frequency Hoping Spread Spectrum version of the physical layer [4]. It also plays an important role in the support of QoS in ad hoc networks, particularly for real-time applications. Furthermore, a common view of local clock time is a basic requirement in some ad hoc routing protocols and cryptography and authentication schemes for detecting out-of-date or duplicated messages.

The high dynamic nature of ad hoc networks, the nondeterminism of the wireless channel and the lack of reference nodes make time synchronization a challenging task in ad hoc networks in that traditional time synchronization techniques for wired networks (e.g., [2]) are no more applicable to ad hoc environment due to their centralized nature and the heavy traffic and computation overhead they involve. A good synchronization mechanism for ad hoc networks should be robust to mobility and topology changes, efficient in terms of traffic and processing cost, scalable and secure.

Here we pay a special emphasis on the security aspect because recently many mechanisms have been proposed to address the time synchronization problem in ad hoc networks [13,14], but most of them do not take into account

* Corresponding author. Tel.: +33 01 4581 7947.
E-mail addresses: Lin.Chen@enst.fr (L. Chen), Jean.Leneutre@enst.fr (J. Leneutre).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.017

security, although security is a major challenge in ad hoc networks. As a result, they are very vulnerable to various attacks ranging from modifying and replaying a time synchronization message to sending forged time values to desynchronize the network or disturb the receiver's clock. Khanna et al. [9] show through simulations that the attacks against IEEE 802.11 Timing Synchronization Function (TSF) can cause significant damage to a large number of nodes. Such insecure time synchronization protocols may further cause serious problems on the applications and protocols based on synchronized time. Nodes may fail to be activated because of the incorrect time estimation, which may cause serious problems such as failing to respond to important events and packet loss. Out-of-date and replayed messages cannot be detected in ad hoc routing protocols or some cryptography and authentication schemes, which may lead to many new opportunities for DoS (Deny-of-Service) and replay attacks.

Given the insecurity of existing time synchronization protocols in ad hoc networks and their detrimental effect on the applications, we propose a novel suite of time synchronization mechanisms for ad hoc networks taking into consideration security, scalability and other challenges. The proposed mechanisms are based on symmetric cryptography, avoiding the costly digital signature schemes which may be undesirable for resource constrained environments such as ad hoc networks.

We start by performing an in-depth analysis on the core problems of existing synchronization mechanisms for ad hoc networks based on which we propose our secure single-hop time synchronization procedure (SSTSP). We show by simulation and analytical studies that SSTSP significantly outperforms existing approaches in terms of scalability, accuracy and security. We then extend our efforts to the secure time synchronization for multi-hop ad hoc networks. We propose the multi-hop secure time synchronization procedure (MSTSP), an extended and adapted version of SSTSP for multi-hop ad hoc networks, and evaluate its performance via simulation. The results show that the performance of MSTSP is significantly superior to TSF and is among the best of currently proposed solutions in terms of accuracy and scalability. Besides, MSTSP can maintain the network synchronized even under malicious attacks.

2. Related work

Time synchronization is the process to ensure that physically distributed processors have a common notion of time. There are two commonly known approaches for time synchronization [1], centralized and distributed. The centralized approach is also known as master–slave synchronization where there is one or more accurate clocks (the master(s)) to which all other nodes listen and adjust their local clocks accordingly. The time synchronization mechanism proposed in Ganeriwal et al. [14] for ad hoc networks belongs to this catalog. The distributed approach is also known as mutual synchronization where there is no master clock, but instead all clocks cooperate to achieve synchronization in a distributed manner. IEEE 802.11 TSF in ad hoc mode belongs to this catalog.

For mobile ad hoc networks we argue that the distributed approach is more suitable due to its robustness, flexibility and adaptability. This motivates us to focus our efforts on the secure distributed time synchronization mechanism for ad hoc networks. IEEE 802.11 TSF is specified by IEEE 802.11 standards as an efficient distributed synchronization mechanism for single-hop ad hoc networks. Other distributed synchronization approaches [11,1] mainly base themselves on IEEE 802.11 TSF and improve it to achieve better performance or extend it to multi-hop ad hoc networks.

2.1. IEEE 802.11 TSF in ad hoc mode

IEEE 802.11 standards specify the ad-hoc-mode Timing Synchronization Function (TSF) for IEEE 802.11 ad hoc networks (IBSS) [4] in which time synchronization is achieved by periodical time information exchange through beacons containing timestamps and other parameters. Each node maintains a local clock counting in increments of microseconds. All nodes in the IBSS compete for beacon transmission every Beacon Period (BP). At the beginning of each BP, there is a beacon generation window consisting of $w+1$ slots each of length *aSlotTime*, where w is a parameter defined by system. Each node calculates a random delay uniformly distributed in $[0,w]\times aSlotTime$ and schedules to transmit a beacon when the delay timer expires. If a beacon is received before the random delay timer has expired, the node cancels the pending beacon transmission. Upon receiving a beacon, the node sets its local clock to the timestamp of the beacon if the value of the timestamp is later than its local clock.

In spite of its distributed nature and its efficiency in terms of communication cost, IEEE 802.11 TSF has the following problems when applied to large scale or multi-hop ad hoc networks:

- *Fastest node asynchronization:* As identified in Lai and Zhou [5], the clock of the fastest node may drift away, because it may not get a chance to transmit its beacon. Since the fastest node does not synchronize itself to other nodes, its clock will keep drifting away from others. The problem becomes more severe when the number of nodes of the network increases.
- *Beacon collision:* As the number of nodes in the network increases, the synchronization beacon transmission contentions uprise accordingly. As a result, in a large network, due to repeated collisions, synchronization beacons can hardly be successfully transmitted and some nodes may fail to synchronize with others.
- *Time partitioning:* As identified in So and Vaidya [6], this problem occurs when TSF is applied to multi-hop ad hoc networks, where nodes fall into clusters which may be desynchronized during a long period of time. When the

number of nodes increases, the problem has a more negative impact on the synchronization accuracy. Giving faster nodes higher chance in beacon transmission will also increases the impact of the time partitioning problem.

2.2. Scalable time synchronization for ad hoc networks

ATSP (Adaptive Timing Synchronization Procedure) was proposed in Lai and Zhou [5] to solve the *fastest node asynchronization* problem in IEEE 802.11 TSF. The basic idea is to let the fastest node compete for beacon transmission every BP and let other nodes compete only every $I_{\max}$ BPs. The parameter $I_{\max}$ should be carefully chosen to reach a tradeoff between scalability and stability. As an improved version of ATSP, the authors propose TATSP (Tiered Adaptive Timing Synchronization Procedure) in which the nodes are dynamically classified into three tiers according to the clock speed. The nodes in tier 1 compete for beacon transmission in every BP and the nodes in tier 2 compete once in a while and the nodes in tier 3 rarely compete. SATSF (Self-adjusting Timing Synchronization Function) is another synchronization protocol proposed in Zhou and Lai [12] compatible with IEEE 802.11 TSF. In SATSF, node i competes for beacon transmission every $FFT(i)$ BPs. $FFT(i)$ is adjusted at the end of each BP in the way that fast nodes will gradually decrease their FFT value, thus competing more frequently than slow nodes.

ASP (Automatic Self-self-time-correcting Procedure) is proposed in Sheu et al. [11] to synchronize multi-hop ad hoc networks. The basic idea is to synchronize the whole network by fulfilling two tasks: to increase the successful transmission probability for faster nodes and to spread the faster time information throughout the whole network. The first task is achieved by increasing the beacon transmission priority of a node who has faster time and by cutting down the priorities of the others. When some slower nodes get enough information to accomplish synchronization by themselves, their beacon transmission priorities are increased to carry out the second task.

Rentel and Kunz propose a mechanism in [1] which differs from the idea of giving faster nodes higher priorities. In the mechanism all nodes participate equally in the synchronization of the network. The authors define a controlled clock, which is an adjusted clock of the real clock, and a parameter $s = \frac{\text{controlled clock}}{\text{real clock}}$. If no beacons are received within last T_DELAY beacons, the node participates the contention with probability p for the next BP. When receiving a beacon, the node updates s and p to synchronize to the sender of the beacon.

2.3. Secure and fault tolerant time synchronization

In spite of the numerous time synchronization protocols proposed for ad hoc networks, most of them have not been built with security in mind. To our knowledge, there exist very few propositions on the secure time synchronization protocols in the literature among which [8] mainly focus on a specific type of attack called delay attack. The authors propose two approaches to detect and accommodate the delay attack. One approach uses the generalized extreme studentized deviate (GESD) algorithm to detect multiple outliers (malicious time offset). The other uses a threshold based on a time transformation technique to filter out the outliers. Ganeriwal et al. [10] proposes several protocols for sensor networks to secure pairwise time synchronization over single hop and multiple hops. The authors further extend their efforts to secure group time synchronization. They propose the lightweight secure group synchronization protocol to counter external attacks and the secure group synchronization protocol to counter both external and internal attacks but at the price of the heavy traffic overhead and the lack of scalability. Sun et al. [19] propose a fault-tolerant cluster synchronization protocol for sensor networks in which the hash chain scheme is applied to achieve local broadcast authentication. All sensors should be initially synchronized to bootstrap the protocol.

Our proposed time synchronization mechanisms differ from existing schemes in the following ways: (1) We take into account the scalability issue in our time synchronization protocols. Our synchronization mechanisms do not operate on synchronization message flooding or exchanging between each pair of nodes which may pose scalability problem; (2) Our synchronization protocols are totally distributed and do not rely on any synchronization sources or hierarchy. This feature makes our approach robust to topology changes and network dynamics in ad hoc environments.

3. System model

3.1. Clock model

Each mobile node is equipped with a clock, a time measurement device normally composed of a hardware oscillator and an accumulator. Mathematically, the measured time $T(t)$ is a function of real time t:

$$T(t) = \int_{t0}^{t} \rho(\tau)\mathrm{d}\tau + T(t_0)$$

where ρ is the nominal frequency of the oscillator, $T(t_0)$ is the initial clock offset.

In an ideally synchronized network, for each node i it holds that $\rho_i(\tau) = 1$ and $T(t_0) = t_0$ for all time long. However, since all hardware clocks are imperfect, the above equations in the ideal case do not hold. As a result, different clocks may drift away from each other. In this paper we use the bounded-drift model in which the difference between $\rho_i(\tau)$ and 1 is bounded by $\Delta\rho_{\max} \sim 10^{-6}$, meaning that the clock drifts away for several seconds in 10 days (10^6 s). Therefore, the synchronization process is indispensable and should be executed periodically. We also assume that during a period of time that is not very long, $\rho_i(\tau)$ does not vary with time. Thus the clock can be regarded as linear with respect to real time during that period of time.

3.2. Time synchronization model

Although time synchronization is essential to many applications in ad hoc networks, the requirement ranges from extreme strict synchronization to loose coordination. Therefore, there are actually many different types of synchronization mechanisms based on different criteria such as the scope and the lifetime of the synchronization [17]. Our mechanism falls into the catalogue of network-wide, server-less, proactive synchronization, as explained as following:

- Network-wide: The proposed synchronization mechanisms provide network-wide synchronization where all nodes within the network achieve approximately the same clock reading. The other end of the spectrum is the synchronization among only a subset of nodes in the network. A typical example is pairwise synchronization that aims to provide synchronization between a pair of neighbor nodes.
- Server-less: Our proposed synchronization mechanisms are fully distributed without any server or external time sources as in NTP [2]. Each node is synchronized with every other node with a time which might be different from the real time. Our approach can meet the need of most applications in ad hoc networks that requires synchronized clock.
- Proactive: The proposed approaches proactive such that the network is maintained synchronized by the repeated execution of the synchronization procedure. In contrast to the proactive synchronization is the on-demand synchronization in which the synchronization procedure is triggered by demand or certain events.

3.3. Attacker model

Attackers may disrupt the operation of the time synchronization protocols by exhibiting malicious behavior: e.g., replay, forge, corrupt synchronization messages to influence the time view of benign nodes, as shown in Table 1. The attacks to the synchronization protocols can be classified as external attacks and internal attacks based on the information the attackers have. External attacks are launched by external attackers who do not have the cryptographic credentials (e.g. public/private key pairs, authenticated hash chains) that are necessary to participate in the synchronization procedure. Internal attacks are launched by internal attackers who have compromised legitimate nodes, and therefore have access to the cryptographic credentials of those nodes. Obviously internal attacks are far more difficult to detect and sometimes cannot be countered by pure cryptographic primitives.

Table 1
Synchronization attacks

Possible attacks to time synchronization protocols
Synchronization message forgery
Synchronization message alteration
Synchronization message replay
Synchronization message relay (delay)
DoS attack

4. Single-hop secure time synchronization procedure

4.1. Design philosophy

Although TSF provides an efficient distributed synchronization mechanism in terms of traffic overhead, it suffers from the scalability problem due to its beacon contention scheme. Besides, it is vulnerable to various malicious attacks. In this section we address the above two problems which are vital to build a scalable and secure synchronization protocol.

First we argue that the root of the scalability problem in TSF lies in the fact that the increase in the number of nodes in the network decreases the synchronization beacon emission opportunity of the fastest node or a subset of the fastest nodes. Some protocols improving TSF increase the successful emission probability of the fastest nodes by attributing them priority with respect to other nodes in the network. These mechanisms are significantly more scalable than TSF, but since they follow the same contention mechanism as TSF, the scalability problem is not totally solved. Furthermore, they usually depend on the observation of the beacons to find and locate the fast nodes, which may increase the latency of synchronization.

SSTSP, however, addresses the scalability problem from another angle. In SSTSP, all nodes content to emit the synchronization beacon at the beginning following the contention mechanism of TSF. The winner becomes the reference node and emits a beacon in the beginning of every BP without random delay. Other nodes synchronize their local clocks to the reference node until the reference node leaves the network, when another round of contention begins. To synchronize to the reference node, a node adjusts its clock parameters to gradually catch up with the pace of the reference clock in order to avoid backward and uncontinuous leaps in time. All nodes have the equal chance to become the reference node, but the contention takes place only when the formal reference node leaves and once the reference node is established, other nodes disable their beacon emission and synchronize their local clocks to the reference node each BP. The proposed mechanism maintains the distributed nature of the synchronization process while removing the scalability problem from its root. By making the full use of every received synchronization beacon and adopting a fine adjustment mechanism, we achieve significantly better synchronization accuracy than TSF and avoid all the backward or other uncontinuous leaps in local adjusted clock. Furthermore, by carefully choosing parameters, our mechanism is robust to the change of the reference node and the loss of synchronization beacons.

Furthermore, our approach can detect malicious synchronization attacks and prevent networks from being

desynchronized by malicious nodes using erroneous time values. To this end, we use μTESLA [18], a lightweight technique base on one-way hash chain, to protect synchronization beacons against external attackers. We would like to mention that traditional security mechanisms based on asymmetric cryptographic operations cannot be applied in our context in that it usually takes up to hundreds of milliseconds depending on the CPU capacity of the nodes, which may increase significantly the synchronization error. Furthermore, due to its nature, nodes in ad hoc networks may be resource constrained. It is sometimes expensive or even prohibited for such nodes to perform heavy asymmetric cryptographic operations. In contrast, hash functions are three to four orders of magnitude faster than asymmetric operations and can be performed in an on-the-fly way such that it causes almost no additional delay.

Moreover, in our approach, we propose the "clock drift check" to counter the internal attacks and other attacks such as delay attacks and replay. This mechanism is based on the fact that the difference between any two clocks cannot drift unboundedly within a certain period of time.

Notations

s_i	Hash chain seed of node i
n	Hash chain length
T_0	Start time of the Hash chain
$h^j(s_i)$	The jth element of node i's Hash chain
BP	Beacon period, typical value is 0.1 s
t_i	Local unadjusted time of node i
$c_i(t)$	Local adjusted time of node i at local time t. Our goal is to synchronize $c_i(t)$
t_i^j	Local unadjusted time of node i when receiving the beacon in the jth BP
t_{ref}^j	Local adjusted time of the reference node when emitting the beacon in the jth BP
k_i^j, b_i^j	Coefficient and offset parameter to be adjusted when receiving the beacon in the jth BP
ρ_i	Nominal frequency of node i's clock, can be regarded as constant during a short period of time
ρ_i'	Nominal frequency of node i's adjusted clock, $\rho_i' = \rho_i * k_i^j$ in the jth BP
t_p	Transmission and propagation delay
T^m	Local expected emission time of the synchronization beacon in the mth BP, $T^m = T_0 + m * BP$ if $m > 1$
ts_{ref}^j	Adjusted timestamp value obtained from the beacon emitted by the reference node in the jth BP. $ts_{\text{ref}}^j = t_{\text{ref}}^j + t_p$. ts_{ref}^j is estimated at the receiver side
$(t_i^j)^*$, $(t_{\text{ref}}^j)^*$,$(ts_{\text{ref}}^j)^*$	Expected values of t_i^j, t_{ref}^j, ts_{ref}^j
ϵ	Maximum error when estimating t_{ref} by ts_{ref}, normally $\epsilon < 5$ μs
σ	Maximum synchronization error in SSTSP
δ	Threshold in the "clock drift check"

4.2. Assumptions and requirements

We assume that each pair of nodes shares a pairwise key which is used to bootstrap the synchronization when a node enters the network. Once the new arriver acquires the initial synchronization, the pairwise keys are no more needed for the following phase.

To use one-way hash chains, we need some mechanism for a node to distribute an authenticated element $h^n(s_i)$ in its hash chain. A traditional approach is to let each node use its public key sign the hash chain element. Alternatively, a node can securely distribute an authenticated hash chain element using pairwise keys [15] or non-cryptographic approaches [16].

We also assume that the synchronization beacons are timestamped below MAC layer. Thus, we remove the most significant non-deterministic factor of the end-to-end delay of the beacons, medium access waiting time.

4.3. Synchronization procedure

4.3.1. Node initiation

Each node i picks a random seed s_i and generates its hash chain based on s_i: $h(s_i)$, $h^2(s_i)$,..., $h^n(s_i)$. The last element $h^n(s_i)$ is authenticated and published within the network. The start time of the hash chain T_0 is also published (e.g. T_0 can be configured and published by the first node arriving in the network or be integrated into the synchronization beacons). Fig. 1 illustrates the Hash chain Scheme. Suppose the beacons are expected to be emitted at time $T_0 + j * BP$ ($j = 1, 2, \ldots, n$). Each element of the hash chain $h^{n-j}(s_i)$ is used as the key to secure the synchronization beacon sent by node i in the time interval $[T_0 + j * BP - BP/2,\ T_0 + j * BP + BP/2]$ if node i is the reference node. Node i, in its synchronization beacon sent in the above time interval, discloses the element $h^{n-j+1}(s_i)$ ($j > 1$), allowing other nodes to authenticate previously received beacons sent by itself in last time interval $[T_0 + (j-1) * BP - BP/2,\ T_0 + (j-1) * BP + BP/2]$.

Our single-hop secure time synchronization procedure (SSTSP) consists of two phases: the bootstrapping phase and the synchronization phase.

4.3.2. Bootstrapping phase

When joining the network, the node first enters the bootstrapping phase during the first BP to acquire the initial synchronization with the rest of the network. This initial synchronization further enables the application of one-way hash chains to secure the time synchronization

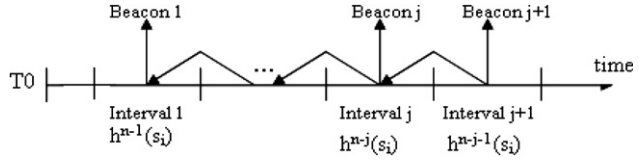

Fig. 1. Hash chain scheme.

procedure in the synchronization phase that follows the bootstrapping phase.

We adopt the following simple pairwise synchronization protocol [10] in this phase:

1. $i(t_i^s) \rightarrow j(c_j^r)$: $i, j, N_i, SynInit$
2. $j(c_j^s) \rightarrow j(t_i^r)$: $c_j^r, c_j^s, N_i, SynAck, MAC_{K_{ij}}\{j(t_i^r): c_j^r, c_j^s, N_i, SynAck\}$
3. If $d = (t_i^r - t_i^s) - (c_j^s - c_j^r) < \beta(T_{init} + T_{ack})$, i sets $t_i = t_i + (c_j^r - t_i^s) - (t_i^r - c_j^s)$

In the above protocol, i is the new arriver that synchronizes to its neighbor j. i sends a SynInit message at local time t_i^s. j receives the message at its local adjusted time c_j^r and send back a SynAck message at c_j^s containing the correspondent timestamps protected by the attached MAC. i then computes the end-to-end delay d and compares d with $\beta(T_{\text{init}} + T_{\text{ack}})$ to check if any packet is delayed or replayed by attacks, where β is a coefficient sightly greater than 1, T_{init} and T_{ack} is the transmission time of the SynInit and SynAck including packet header and preamble. In normal cases, we have $d \sim T_{\text{init}} + T_{\text{ack}}$ since the propagation time is negligible and the calculation of keyed MAC can be processed in the on-the-fly as the packet is being transmitted. If the SynInit or SynAck packet is replayed or delayed by an attacker, then since the attacker cannot receive and emit at the same time, we have $d \geqslant 2(T_{\text{init}} + T_{\text{ack}})$. Next i computes the offset between its local clock and j's clock and uses the offset to adjust its clock.

i thus synchronizes itself with a trusted neighbor j or repeats the above protocol with several neighbors if it does not have any trusted neighbors and further eliminates biased offsets and uses the averaged of the rest unbiased offset to adjust its local clock. In the bootstrapping phase, $c_i(t_i)$ is set to t_i. At the end of the bootstrapping phase, i is synchronized with the network and then enters the following synchronization phase.

4.3.3. Synchronization phase

In this phase, each node competes to be the reference node if it has not heard the synchronization beacon in the last l BPs. A larger value of l makes the mechanism more robust since the failure to receive a beacon may be due to collision or temporary wireless channel instability other than the leave of the reference node. As price, a larger l increases the synchronization error when the reference node changes. In case of collision, the contention may last several BPs. The contention mechanism is the same as in IEEE 802.11 TSF. The winner becomes the reference node and emits a beacon in the beginning of every BP without random delay. Other nodes synchronize their local clocks to the reference node until the reference node leaves the network, when another round of contention begins. A node joining the network does not participate in the contention until it is synchronized with the network. We use μTESLA scheme to protect the beacons. The synchronization beacon sent by the reference node ref in time interval j is:

$$\langle B, j, h_{\text{ref}}^{n-j}(B, j), h^{n-j+1}(s_{\text{ref}}) \rangle$$

where B is the original unsecured synchronization beacon, $h_{\text{ref}}^{n-j}(B, j)$ denotes the HMAC output using $h^{n-j}(s_{\text{ref}})$ as the key applied to (B,j), $h^{n-j+1}(s_{\text{ref}})$ is the disclosed key corresponding to the last interval (interval $(j-1)$).

Each node i temporarily stores the recently received beacons. Upon receiving a new beacon from the reference node ref, node i performs the following checks:

- Node i checks whether interval j corresponds to the current time interval.
- If the above check passes, node i further checks the validity of the disclosed key $h^{n-j+1}(s_{\text{ref}})$ in the beacon by verifying whether $h^{j-1}(h^{n-j+1}(s_{\text{ref}}))$ equals to the published element $h^n(s_{\text{ref}})$. In case of success, i checks the authenticity and the integrity of the beacon received in last interval using disclosed key $h^{n-j+1}(s_{\text{ref}})$. Node i can store previously authenticated disclosed key $h^{n-j+2}(s_{\text{ref}})$ to reduce processing overhead. In this case only one hash operation is needed instead of $j-1$.
- If the above two checks pass, i performs the "clock drift check" whether $|ts_{\text{ref}}^j - c_i(t_i^j)| < \delta$, where the threshold δ is the bound of the clock drift between $c_i(t_i^j)$ and ts_{ref}^j. In case where i has not received beacons during last l BPs because the reference node changes, $\delta = (l+2)\sigma$.[1] If i has just entered the synchronization phase, $\delta = \sigma + 4\Delta\rho_{\max}BP$.[2] In other cases, $\delta = \sigma$. If the check fails, the beacon may be replayed or delayed or the timestamp is forged by an internal attacker.

If all the above tests pass, node i then adjusts its local clock using the authenticated beacon $(j-1)$ and $(j-2)$. Note that beacon j cannot be used for clock adjustment until its integrity is verified. In SSTSP each node i has two clocks: an original clock and an adjusted clock. The original clock is the hardware clock of the node, e.g. a 64-bit counter with the resolution of 1 μs in the IEEE 802.11 standard. The adjusted clock takes t_i, the time of the original clock as input and adjusts its value $c_i(t_i)$ according to the following relation:

$$c_i(t_i) = k_i^j * t_i + b_i^j, \quad j = 1, 2, \ldots \tag{1}$$

Our objective is to synchronize the adjusted clocks of all the nodes in the network by repeatedly adjusting k_i^j and b_i^j ($k_i^j = 1$, $b_i^j = 0$ if $j \leqslant 2$) at each node when receiving the beacon in the jth BP from the reference node. Compared with IEEE 802.11 TSF, SSTSP has the following desirable features:

[1] See Lemma 2 for the proof that after the reference node changes, the synchronization error after the change is $(l+2)$ times as much as the synchronization error before the change.

[2] It is easy to show that the maximum clock drift during one BP is bounded by $2\Delta\rho_{\max}BP$. When entering the synchronization phase, the difference between the clock of i and the reference node increases in the beginning $2BP$s before i can adjust its clock using authenticated beacons.

- SSTSP achieves better accuracy than IEEE 802.11 TSF via a more sophisticated adjustment scheme in which both the offset and the coefficient parameters are adjusted.
- There is no backwards or other uncontinuous leaps in local clock. This feature is important in some applications. IEEE 802.11 TSF only guarantees that no backwards leaps exist.

The following equations illustrate the clock adjustment of SSTSP:

$$k_i^{j-1} * t_i^j + b_i^{j-1} = k_i^j * t_i^j + b_i^j \tag{2}$$

$$c_i((t_i^{j+m})^*) = k_i^j * (t_i^{j+m})^* + b_i^j = (ts_{\text{ref}}^{j+m})^* \tag{3}$$

$$\frac{t_i^{j-1} - t_i^{j-2}}{ts_{\text{ref}}^{j-1} - ts_{\text{ref}}^{j-2}} = \frac{(t_i^{j+m})^* - t_i^{j-1}}{(ts_{\text{ref}}^{j+m})^* - ts_{\text{ref}}^{j-1}} \tag{4}$$

$$(ts_{\text{ref}}^{j+m})^* = T^{j+m} \tag{5}$$

Eq. (2) follows the argument that the adjusted clock $c_i(t_i)$ is continuous at t_i^j. Eq. (3) indicates that the adjusted clock of node i is expected to converge to the reference clock at the expected receiving time of the beacon $(j+m)$. Before the convergence, the synchronization error is expected to decrease monotonously. m $(m>1)$ is the parameter of aggressiveness. A larger value of m increases the synchronization latency since the local clock converges slower to the reference node, while it avoids the synchronization error to be increased significantly when the reference node changes. Eq. (4) establishes the relation of the local clock and the adjusted clock of the reference node based on the linearity of the clocks. As discussed in Section 3.1, the original clock is regarded as a linear function of real time within a short period of time. The adjusted clock is regarded as linear as long as no adjustment occurs during that period of time. Eq. (5) follows that the expected emission time of the $(j+m)$th beacon is T^{j+m}. By solving Eqs. (2), (3), (4) and (5) containing 4 variables k_i^j, b_i^j, $(ts_{\text{ref}}^{j+m})^*$ and $(t_i^{j+m})^*$, we get:

$$k_i^j = \frac{(T^{j+m} - (k_i^{j-1} * t_i^j + b_i^{j-1})) * (ts_{\text{ref}}^{j-1} - ts_{\text{ref}}^{j-2})}{(t_i^{j-1} - t_i^{j-2}) * (T^{j+m} - ts_{\text{ref}}^{j-1}) + (t_i^{j-1} - t_i^j) * (ts_{\text{ref}}^{j-1} - ts_{\text{ref}}^{j-2})}$$

$$b_i^j = k_i^{j-1} * t_i^j + b_i^{j-1} - \frac{(T^{j+m} - (k_i^{j-1} * t_i^j + b_i^{j-1})) * (ts_{\text{ref}}^{j-1} - ts_{\text{ref}}^{j-2}) * t_i^j}{(t_i^{j-1} - t_i^{j-2}) * (T^{j+m} - ts_{\text{ref}}^{j-1}) + (t_i^{j-1} - t_i^j) * (ts_{\text{ref}}^{j-1} - ts_{\text{ref}}^{j-2})}$$

By repeatedly updating k_i^j and b_i^j using received beacons from the reference node, the local adjusted clock of each node i gradually catches up with the pace of the reference clock and the network is hence synchronized.

4.4. Effectiveness of SSTSP

In this section, we provide analytical analysis on the effectiveness of SSTSP by studying the synchronization error bound of SSTSP. Lemma 1 shows that regardless of the initial value, the adjusted clock of i, c_i, converges to the adjusted timestamp of the reference node ts_{ref}, where $ts_{\text{ref}} = t_{\text{ref}} + t_p$.

Lemma 1. *For any node i, its local adjusted clock, c_i, converges to ts_{ref}.*

Proof. Let D_i^n be the difference between $c_i(t_i^n)$ and ts_{ref}^n when receiving the nth beacon: $D_i^n = c_i(t_i^n) - ts_{\text{ref}}^n$. Let $(n+q) * BP + d_{n+q}$ be the time when the reference node emits the $(n+q)$th beacon $(q \geqslant 1)$, where d_{n+q} is the time elapsed between the scheduled emission time of the beacon and its actual emission time. The timestamp in the beacon is adjusted at node i by adding t_p to ts_{ref}^{n+q}: $ts_{\text{ref}}^{n+q} = (n+q) * BP + d_{n+q} + t_p$.

By (3) we have:

$$k_i^n * (t_i^{n+m})^* + b_i^n = (ts_{\text{ref}}^{n+m})^* = (n+m) * BP + t_p \tag{6}$$

Apply (1) at t_i^n and t_i^{n+1} we get:

$$c_i(t_i^n) = k_i^n * t_i^n + b_i^n = ts_{\text{ref}}^n + D_i^n = n * BP + d_n + t_p + D_i^n \tag{7}$$

$$c_i(t_i^{n+1}) = k_i^n * t_i^{n+1} + b_i^n = ts_{\text{ref}}^{n+1} + D_i^{n+1} = (n+1) * BP + d_{n+1} + t_p + D_i^{n+1} \tag{8}$$

By the linearity of the clock we have:

$$\frac{(t_i^{n+m})^* - t_i^n}{(ts_{\text{ref}}^{n+m})^* - ts_{\text{ref}}^n} = \frac{(t_i^{n+m})^* - t_i^{n+1}}{(ts_{\text{ref}}^{n+m})^* - ts_{\text{ref}}^{n+1}} \tag{9}$$

Combining (6)–(9), we get:

$$\frac{D_i^{n+1}}{D_i^n} = \frac{(m-1) * BP - d_{n+1}}{m * BP - d_n} < \begin{cases} \frac{d}{m * BP - d} & m = 1 \\ \frac{(m-1) * BP}{m * BP - d} & m > 1 \end{cases}$$

where $d = \max(d_j)$, $(j > 1)$. Recursively we get:

$$\frac{D_i^{n+q}}{D_i^n} < \begin{cases} (\frac{d}{m * BP - d})^q & m = 1 \\ (\frac{(m-1) * BP}{m * BP - d})^q & m > 1 \end{cases}$$

Given any synchronization error threshold Δ and D_i^n, after

$$[\log_{\frac{d}{(m * BP - d)}} \frac{\Delta}{D_i^n}] \text{ BPs} \quad (\text{if } m = 1) \text{ or}$$

$$[\log_{\frac{(m-1) * BP}{(m * BP - d)}} \frac{\Delta}{D_i^n}] \text{ BPs} \quad (\text{if } m > 1)$$

the difference between the local adjusted clock of node i and the clock of the reference node will drop below the threshold. The adjusted clock thus converges to ts_{ref}. □

Apply Lemma 1 and $|ts_{\text{ref}} - t_{\text{ref}}| < \epsilon$, it is easy to prove that the maximum synchronization error is bounded by 2ϵ, typically 10 μs.

Lemma 2 studies the difference between c_i and ts_{ref} when the reference node changes.

Lemma 2. *For any node i, let D_i^- and D_i^+ be the difference between c_i and ts_{ref} (ref is the old reference node) before and after the reference node changes, then $D_i^+ < (l+2) * D_i^-$.*

Proof. It takes $(l+3)$ BPs before node i can re-adjust its local clock to the new reference clock: during the first $(l+1)$ BPs, the new reference node is elected via contention; during the following two BPs, each node validates the timestamp sent by the new reference node in previous BP and gets enough validated timestamps to adjust its local clock. Let $t_{\text{ref}}^n = n * BP + d_n$ (d_n is defined the same as in Lemma 1) be the time when the last beacon is emitted by the old reference node. The beacon is received by i at local unadjusted time t_i^n. The difference between $c_i(t_i^n)$ and ts_{ref}^n is D_i^-. Let $t_{\text{ref}}^{n+l+3} = (n+l+3) * BP + d_{n+l+3}$ be the local time of the old reference node when the new reference node emits its beacon in the $(l+3)$th BP with which node i begins to adjust its local clock to the new reference clock. The difference between $c_i(t_i^{n+l+3})$ and ts_{ref}^{n+l+3} is D_i^+. Between ts_{ref}^n and ts_{ref}^{n+l+3}, the synchronization error cannot be controlled since no adjustment is done during this period. After ts_{ref}^{n+l+3} all the nodes synchronize to the new reference node, and the synchronization error decreases. We prove in the following that $D_i^+ < (l+2) * D_i^-$.

By (3) we get:

$$k_i^n * (t_i^{n+m})^* + b_i^n = (ts_{\text{ref}}^{n+m})^* = (n+m) * BP + t_p \tag{10}$$

Apply (1) at t_i^n and t_i^{n+l+3} we get:

$$c_i(t_i^n) = k_i^n * t_i^n + b_i^n = ts_{\text{ref}}^n + D_i^- = n * BP + d_n + t_p + D_i^- \tag{11}$$

$$c_i(t_i^{n+l+3}) = k_i^n * t_i^{n+l+3} + b_i^n = ts_{\text{ref}}^{n+l+3} + D_i^+ = (n+l+3) * BP + d_{n+l+3} + t_p + D_i^+ \tag{12}$$

By the linearity of the clock we have:

$$\frac{(t_i^{n+m})^* - t_i^n}{(ts_{\text{ref}}^{n+m})^* - ts_{\text{ref}}^n} = \frac{(t_i^{n+m})^* - t_i^{n+l+3}}{(ts_{\text{ref}}^{n+m})^* - ts_{\text{ref}}^{n+l+3}} \tag{13}$$

Combining (10)–(13), we get

$$\frac{D_i^+}{D_i^-} = \frac{(m-l-3)BP - d_{n+l+3}}{mBP - d_n}$$

Note that d_n, $d_{n+l+3} \ll BP$, we get

$$\frac{D_i^+}{D_i^-} = \frac{m-l-3}{m} + o(1)$$

We can see from the proof that the optimal value of m in terms of the performance when the reference node changes is $l+3$ in that the adjusted clock of each node is expected to converge to the same time when a new round of synchronization begins. Even in the worst case where $m=1$, D_i^+ can be bounded by $(l+2) * D_i^-$. □

It is further easy to prove that the synchronization error after the change of the reference node is bounded by $|\frac{m-l-3}{m}| * syn + 2\epsilon$, where syn_err is the synchronization error before the reference node changes. Combining Lemma 1 and Lemma 2, we have the following theorem on the synchronization error of SSTSP:

Theorem 1. *The synchronization error of SSTSP σ can be bounded by $|\frac{m-l-3}{m}| \cdot 2\epsilon + 2\epsilon$.*

From the above theorem, we can see that by carefully configuring the parameters, the synchronization of SSTSP can be controlled under 10 μs.

4.5. Traffic and storage overhead

In terms of traffic overhead, the number of synchronization beacons emitted in SSTSP is the same as in TSF, while the size of each beacon increases from 56 bytes (24 bytes of preamble and 32 bytes of data) in TSF to 92 bytes (assume 128-bit hash values are used) in SSTSP due to the hash values and the interval index included to secure the beacons.

Besides, each node is required to store its own hash chain. It can either create the hash chain all at once and store all the elements or only store the last element and compute the new element on demand. Jakobsson [7] proposes a hybrid storage efficient mechanism to reduce storage with a small recomputation penalty: a one-way hash chain with n elements only requires $\log_2(n)$ storage and $\log_2(n)$ computation to access an element. Each node is also required to buffer temporarily the synchronization beacons received during last 2 BPs. In most cases 300–500 bytes of memory can meet the requirement. We argue that the storage requirement as well as the increase in the beacon size is reasonable considering the gain in performance and security that SSTSP achieves.

4.6. Security analysis

Synchronization beacon forgery and alteration: Attackers may attack the synchronization protocols by forging or modifying synchronization beacons. SSTSP uses the μTESLA scheme to ensure the integrity and authenticity of the synchronization beacons. This prevents the external attackers from modifying or forging the synchronization beacons or impersonating the reference node. A more serious case is when an internal attacker becomes the reference node. In this case, the guard time check serves as a defense line to decrease the effectiveness of the attacks such that the attacker can only forge timestamps whose difference with the receiver's local time is within the guard time, otherwise the beacons containing incorrect timestamps are rejected. We argue that the impact of this attack is limited in that all nodes are synchronized to a virtual clock that may be slightly different to the real clock of the reference node. However, the internal attacker cannot desynchronize the network.

Synchronization beacon replay and relay: Attackers may replay the out-of-date synchronization beacons to deliberately magnify the offset of the time declared in the replayed message and actual time. As a more delicate version of replay attacks, an attacker may firstly jam the channel between the reference node and the victim node A, then delay the synchronization beacons from the reference node and relay it to A later to make A incorrectly estimate the time of the reference node (This attack is referred to as pulse-delay attack in Ganeriwal et al. [10]). The "clock drift check" can thwart these attacks. The argument is that in most cases ($l < 4$), if the beacon is replayed or relayed, since the attacker cannot receive and emit at the same time, we have $|ts_{\text{ref}}^j - c_i(t_i^j)| > T_{\text{beacon}} > \delta$, where T_{becon} is the trans-

mission time of a beacon including the header and preamble (note that transmitting only the short preamble requires 96 μs in IEEE 802.11b, even in IEEE 802.11g ERP-OFDM, we have $T_{\text{beacon}} = 50.7\ \mu s > \delta$). Even in extreme cases where $l \geqslant 4$, we can add a number of bits as padding in the beacons (the padding is also included as the HMAC input) to counter the replay or relay attacks such that after padding, we have $T_{\text{beacon}} > \delta$.

Deny of service: Besides the efforts to violate the proper behavior of the synchronization procedure, attackers can disturb the transmission of synchronization beacons at the beginning of each BP or simply generate a massive amount of messages to jam the wireless channel, impeding the traffic including the transmission of time synchronization beacons. Jamming attacks are beyond the scope of our discussion. Actually under jamming attacks all communications in the network are disabled.

4.7. Performance analysis

In this section, we analyze the influence of the following factors on the synchronization protocols.

- Medium access delay: the waiting time at MAC layer before accessing the channel. This delay is non-deterministic in nature ranging from a few microseconds to a few minutes. In SSTSP, timestamping the beacons below MAC layer removes this delay, the most significant factor in the synchronization process.
- Beacon transmission delay: time taken in transmitting the beacon bit-by-bit at the radio of the sender node. This delay is hundreds of microseconds and deterministic in nature depending on the beacon size and the radio speed. So and Vaidya [6] show that the beacon transmission delay only adds a few nanoseconds to the synchronization error. Its influence can be annulled by taking into account the delay when adjusting the local clock.
- Beacon propagation delay: time over the wireless link between the sender and receiver node. This delay is typically less than 1 μs.

4.8. Simulation study

We further evaluate the performance of SSTSP by simulation. We set the relative clock frequency with respect to real time uniformly distributed in the range of $[1 - 0.01\%, 1 + 0.01\%]$, which is the worst clock accuracy allowed by the 802.11 standards. We run the simulation for 1000 s for OFDM system with bit-rate of 54 Mbps: $w = 30$, $BP = 0.1$ s, $l = 1$, the number of nodes $N = 100 - 500$ and the beacon length is 4 slot time. We also set the packet error rate to be 0.01%. We let 5% of the stations leave at BP $k * 200$ s ($k > 1$). They return after 50 s. In order to simulate the impact of changing the reference node, we let the reference node leave at 300 s, 500 s and 800 s.

IEEE 802.11 TSF: Figs. 2 and 3 show the maximum clock drift of IEEE 802.11 TSF in the network of 100 and 300 nodes. We can see the scalability problem due to the *fastest node asynchronization* and the *beacon collision* problem discussed in Section 2.

SSTSP: Figs. 4 and 5 show the maximum clock drift of SSTSP in the network of 500 nodes with m = 4 and m = 1. We can see that SSTSP significantly outperforms IEEE 802.11 TSF by achieving a very precise synchronization with the maximum clock difference below 15 μs after the protocol stabilizes, which is among the best results of currently proposed solutions (see [11,12] for their detailed results). Comparing the two figures, we can see that a larger m achieves better performance when the reference node changes. Table 2 studies the maximum clock difference of different m. We suggest to choose $m = 2$ or 3 to reach a better tradeoff between synchronization accuracy and synchronization latency.

Performance under attacks: We also simulate IEEE 802.11 TSF and SSTSP in a hostile environment where an attacker attacks the synchronization protocols during

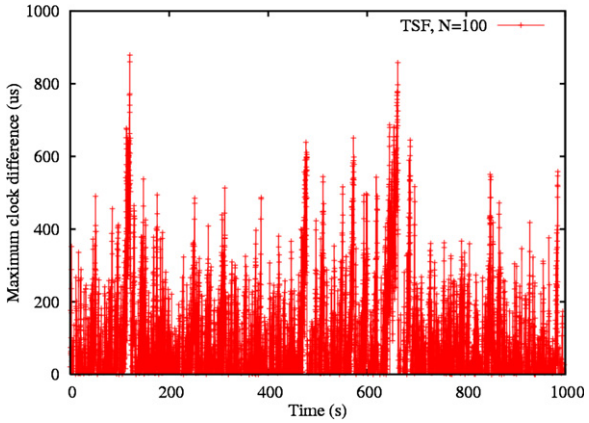

Fig. 2. Maximum clock difference: IEEE 802.11 TSF, 100 nodes.

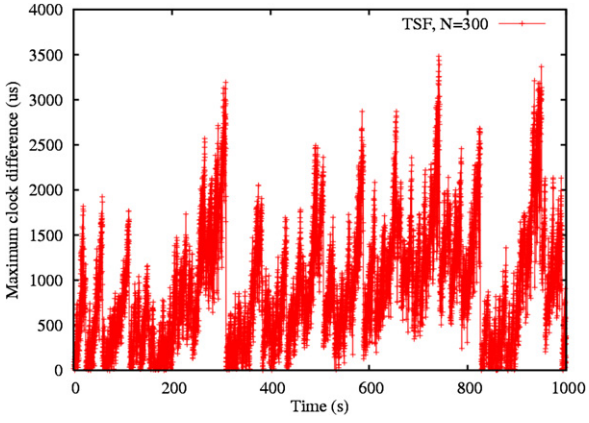

Fig. 3. Maximum clock difference: IEEE 802.11 TSF, 300 nodes.

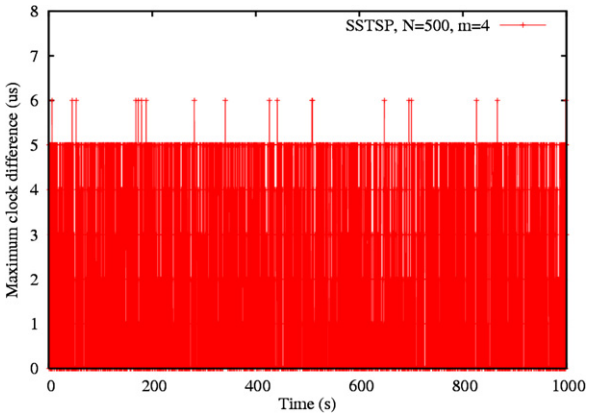

Fig. 4. Maximum clock difference: SSTSP, 500 nodes, $m = 4$.

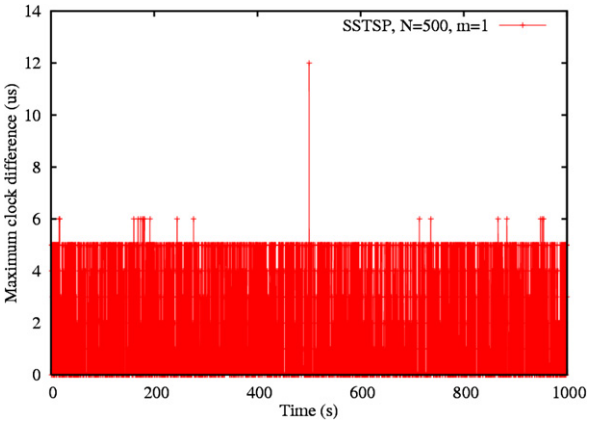

Fig. 5. Maximum clock difference: SSTSP, 500 nodes, $m = 1$.

Table 2
Maximum clock difference vs m

m	Maximum clock difference (μs)
1	12
2	7
3	6
4	6
5	6

400–600 s. The attacker attacks by deliberately sending the synchronization beacons at each BP without delay with an erroneous time value slower than its local clock. We carefully configure the erroneous time values such that they can pass the guard time check in SSTSP. Figs. 6 and 7 show the synchronization error of IEEE 802.11 TSF and SSTSP under the above attack. The synchronization error of IEEE 802.11 TSF uprises to 20,000 μs during the attack. The attacker always wins the contentions thus disabling the fast nodes from emitting beacons. Other protocols improving IEEE 802.11 TSF are also vulnerable to the attack because they depend on the fast nodes to spread the timing information. However, in SSTSP the attacker cannot desynchronize the network even though it manages to become the reference.

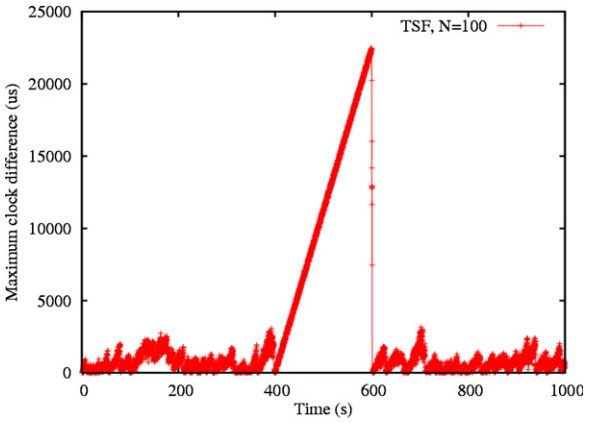

Fig. 6. Maximum clock difference: IEEE 802.11 TSF, 100 nodes, an attacker.

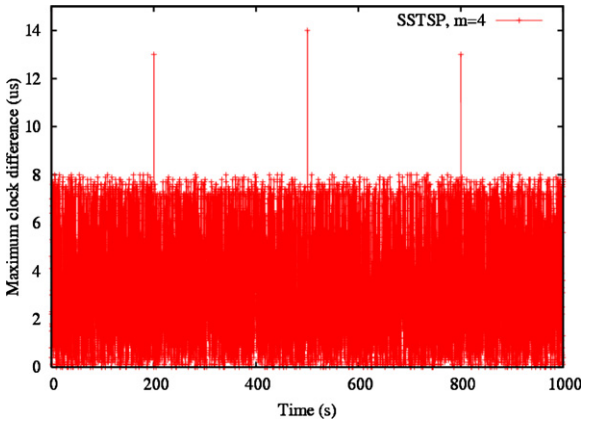

Fig. 7. Maximum clock difference: SSTSP, 500 nodes, an attacker.

5. Multi-hop secure time synchronization procedure

In the rest of the paper we consider a more challenging task, secure and scalable time synchronization in multi-hop ad hoc networks. Compared with many existing synchronization protocols that form a synchronization tree in the network, our secure multi-hop time synchronization procedure (MSTSP) is fully distributed and server-less. The synchronization is done only locally, without a global synchronization leader. This feature makes MSTSP robust to topology change and link failure, which happen frequently in multi-hop mobile ad hoc networks.

5.1. Overview

In MSTSP, we extend SSTSP, our secure synchronization mechanism for single-hop networks presented in Section 4, to multi-hop networks by fulfilling the following two tasks:

First, we divide the multi-hop network into single-hop clusters by running SSTSP. The clusters are thus automatically formed and are overlapped by nature. A random delay is added before each synchronization beacon emission to

avoid the collision with the beacon emission of neighbor clusters. The intra-cluster synchronization is achieved by SSTSP and each node in the overlapping area randomly chooses the cluster reference node as its reference node and synchronizes to it before it moves out of the transmission range.

We then synchronize all cluster reference nodes to achieve network-wide synchronization. We base our design on the topology redundancy, an intrinsic nature of ad hoc networks that can provide certain level of tolerance to potential attacks. More specifically, we make use of nodes in the overlapping area that belong to more than one clusters (we refer to them as bridge nodes) to relay timestamps among the cluster reference nodes. After collecting the timestamps of the neighbor cluster reference nodes, each cluster reference node synchronizes itself to the fastest neighbor cluster reference node. We take the advantage of the redundancy of ad hoc networks to secure the inter-cluster or network-wide synchronization. In MSTSP, the *time partitioning* problem is avoided by the periodical exchange of time information among neighbor clusters. As a result, the network synchronizes to the fastest cluster reference node.

We give a simple motivating example to illustrate the time synchronization between 2 neighbor cluster reference nodes in MSTSP. Consider Fig. 8 where node r_1 and r_2 are two neighbor cluster reference nodes with 3 bridge nodes A, B and C in the overlapping area. Assume r_1 needs to synchronize to r_2 because its clock is slower than that of r_2. Suppose A receives the synchronization beacons from r_1, r_2 at original time t_1, t_2 containing timestamps T_1, T_2. r_1 can estimate the time difference between its adjusted clock and that of r_2 using the original time of A as reference: $\Delta t_{12} = (T_2 - t_2) - (T_1 - t_1)$.

Two points worth further explanation: (1) the adjusted clock of A cannot be used as reference because it may be adjusted between t_1 and t_2. (2) actually Δt_{12} is measured in local original clock of A, the real time difference measured in the adjusted clock of r_1 is $\Delta t'_{12} = k_1^j \frac{\rho_1}{\rho_A} \Delta t_{12}$. We have $|\Delta t'_{12} - \Delta t_{12}| = |(k_1^j \frac{\rho_1}{\rho_A} - 1)\Delta t_{12}|$. Given that $k_1^j \sim 1$, $\frac{\rho_1}{\rho_A} \sim 1$ and $\Delta\ t_{12}$ is in order of 10^1–10^2 μs, we can ignore the difference between $\Delta t'_{12}$ and Δt_{12} and use $\Delta\ t_{12}$ as the estimation of time difference since $\Delta t'_{12}$ cannot be calculated without the knowledge of ρ_A and ρ_1.

To ensure the integrity of the time values t_1, t_2, T_1, T_2, A puts them in the $t_$ex (time exchange) message: $\langle SB_1, t_1, SB_2, t_2, h_A^{n-j}(t\_\text{ex}), h_A^{n-j+1}(s_A)\rangle$, where SB_1, SB_2 are received synchronization beacons from r_1, r_2 containing T_1, T_2, $h_A^{n-j}(t\_\text{ex})$ is the HMAC outputs applied to the message with the Hash chain element of A corresponding to current time interval as key, $h_A^{n-j+1}(s_A)$ is the disclosed key for last time interval.

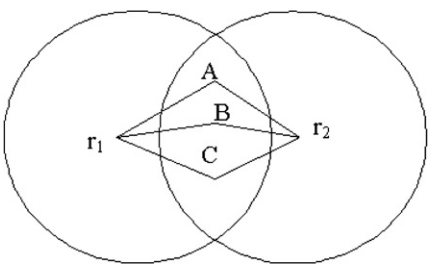

Fig. 8. Inter-cluster synchronization.

It is possible that A is compromised and thus may forge t_1 and t_2. To counter this attack, r_1 estimates Δt_{12} via different bridge nodes A, B, C. r_1 then eliminates the biased values and sets Δt_{12} to the mean of the rest unbiased values. As a result, if only one of the three nodes is compromised, its impact on the synchronization can be removed. To avoid the collision of $t_$ex message transmission, each bridge node should desynchronize its $t_$ex emission (e.g., at time $T_0 + n * BP + BP/2 + d$, where d is randomly chosen in $[0, d_{\max}]$). To make MSTSP adaptive to ad hoc networks of different density, each bridge node emits $t_$ex messages with a pre-configured probability P_s.

5.2. MSTSP

Based on the above analysis, we add the following mechanisms to extend SSTSP to multi-hop networks and build our multi-hop secure time synchronization procedure (MSTSP).

- The reference nodes wait a random delay in the range $[0, w] \times aSlotTime$ before emitting the synchronization beacon at each BP to avoid the collision with the reference nodes in neighbor clusters. If a synchronization beacon is heard during the waiting time, the node stop the pending beacon transmission and synchronizes itself to the sender of the beacon.
- We define a parameter P_s for the bridge nodes as the probability of transmitting the $t_$ex message in each BP. P_s depends on the density of the network and a larger value of P_s increases the robustness of MSTSP to internal attacks at the price of higher traffic overhead. Each bridge node b picks a random number *rand* from $[0, 1]$ in each BP and compares *rand* with P_s. If $rand < P_s$, it transmits $t_$ex containing all the synchronization beacons received from the cluster reference nodes during the last BP as well as the timestamps of the local original time when it receives them.

$$t\_\text{ex} : \langle SB_1, t_1, SB_2, t_2, \ldots, SB_n, t_n, h_b^{n-j}(t\_\text{ex}), h_b^{n-j+1}(s_b)\rangle$$

where SB_i is the synchronization beacon emitted by the cluster reference node i, t_i is the local original time when receiving SB_i containing timestamp T_i, $h_b^{n-j}(t\_\text{ex})$ is the HMAC applied to the whole message with the Hash chain element of b corresponding to current time interval as key, $h_b^{n-j+1}(s_b)$ is the disclosed key used in last interval. The transmission is scheduled at $T_0 + n * BP + BP/2 + d$ to asynchronize the transmission of other bridge nodes, where d is randomly chosen in $[0, d_{\max}]$.

If a bridge node detects that the difference between any two reference nodes i and j is beyond certain threshold by checking $|(T_i - t_i) - (T_j - t_j)| > 2\beta\Delta\rho_{\max}BP$ (β is the

tolerant coefficient slightly greater than 1), it notifies other nodes by flooding an signed alert with received beacons as proof. If multiple bridge nodes detect the abnormal difference, the synchronization process is re-initiated.

- Each cluster reference node A collects the synchronization beacons of the neighbor cluster reference nodes via bridge nodes and synchronizes itself to the fastest neighbor reference node by performing the following operations:

1. For the bridge nodes that send $t_$ex message in both current BP and last BP, A uses the disclosed keys in $t_$ex messages received in current BP to verify the authenticity and integrity of the $t_$ex messages received in last BP. Moreover, since the received $t_$ex messages contain the beacons of neighbor cluster reference nodes, A can use the disclosed keys in the beacons contained in $t_$ex messages received in current BP to verify the beacons contained in $t_$ex messages received in last BP. Note that A cannot directly get the disclosed keys of its neighbor cluster reference nodes since A cannot hear them. However, A can obtain them via bridge nodes in that the synchronization beacons are included in the $t_$ex messages. By doing so, A thus collect a number of verified $t_$ex message containing verified beacons sent by neighbor cluster reference nodes.
2. A then uses the timestamps in these verified messages to synchronize its local adjusted clock to the adjusted clock of the fastest cluster reference node as illustrated in the example. To this end, A collects all the timestamps in the verified $t_$ex messages containing verified synchronization beacons among which T_r, t_b^r, T_A, t_b^A are, respectively, the timestamp in the beacon of the cluster reference r, the original time when the bridge node b receives the beacon from r, the timestamp in the beacon of A, the original time when b receives the beacon from A. A then computes $\Delta t_b^r = (T_r - t_b^r) - (T_A - t_b^A)$ which indicates the approximate time difference between A and r in last BP estimated via b. By collecting the time difference Δt_r^b from different bridge nodes b and eliminating the outliers, A thus obtains the estimated time difference with r, Δt_r by averaging them. A then picks $\Delta t^* = \max(\Delta t_r)$, the largest value from the time differences with all neighbor cluster reference node r and adds Δt^* to its original clock t_A and adjusted clock c_A if $\Delta t^* > 0$.

In the following lemma, we show that in the stabilized case, SSTSP is applicable in MSTSP to achieve intra-cluster synchronization.

Lemma 3. *SSTSP is applicable in MSTSP to achieve intra-cluster synchronization in the stabilized case.*

Proof. In the stabilized case, each cluster reference node A synchronizes via a synchronization route composed of a suite of cluster reference nodes to the fastest cluster reference node r_n: $A, r_1, r_2, \ldots, r_{n-1}, r_n$. On this route, r_{i-1} adjusts its clock once each BP according to the timestamps in the beacons emitted by r_i. Since the local adjusted clock of r_n can be regarded as linear in that no adjustment is performed at the fastest cluster reference node in MSTSP, the adjusted clock of r_{n-1} can be regarded as linear because r_{n-1} adjusts its clock c_{n-1} by adding approximately the same time difference $(\rho'_n - \rho'_{n-1}) * BP$ in each BP. Recursively we can prove that the local adjusted time of A, c_A, can be regarded as linear. This makes (4) hold and justifies that SSTSP can be applied in the multi-hop case for intra-cluster time synchronization. In reality, the mobility of the nodes and the dynamic nature of clusters increase the synchronization error of intra-cluster in a multi-hop network. □

Based on Lemma 3, we have the following result on the effectiveness of MSTSP

Theorem 2. *In stabilized cases, the synchronization error of MSTSP can be bounded by* $2NBP\Delta\rho'_{max} + 2\epsilon$, *where N is the upper bound of the synchronization route (in number of hops) mentioned in Lemma 3,* $\Delta\rho'_{max} = max(\rho'_i - \rho'_j) \sim 10^{-6}$.

Proof. We consider node i who synchronizes via a synchronization route R to the fastest cluster reference node r_n: $r_1, r_2, \ldots, r_{n-1}, r_n$, $1 \leqslant n \leqslant N+1$. In an m-hop ad hoc network, we have $N \sim O(m)$. We now study the synchronization error between two reference nodes in neighbor clusters r_j, r_{j+1}. According to MSTSP, in the k^{th} BP, r_j receives the synchronization beacon of r_{j+1} via bridge nodes containing the disclosed hash element with which it checks the integrity and authenticity of the beacon sent by r_{j+1} in the $(k-1)$th BP and adjusts its clock accordingly, thus the difference of the adjusted clock of r_j and r_{j+1} can be bounded by $2BP\Delta\rho'_{max}$: $|c_j(t) - c_{j+1}(t)| < 2BP\Delta\rho'_{max}$. Recursively we get $|c_1(t) - c_n(t)| < 2nBP\Delta\rho'_{max}$. Since r_n is the fastest reference nodes, we have $0 < c_1(t) - c_n(t) < 2nBP\Delta\rho'_{max}$. Apply Lemma 1, we have $|c_i(t) - c_1(t)| < \epsilon$. Therefore, we have $2nBP\Delta\rho'_{max} - \epsilon < c_i(t) - c_n(t) < 2nBP\Delta\rho'_{max} + \epsilon$, which holds for each node in the network. For any two nodes A and B, apply the above inequality, we get $|c_A(t) - c_B(t)| < 2NBP\Delta\rho'_{max} + 2\epsilon$. □

5.3. Security analysis

In Section 4.6, we have performed the security analysis of SSTSP, showing that SSTSP is powerful enough to prevent single-hop networks or clusters from being desynchronized by both malicious external and internal attackers. In this section we focus on the attack resistance of inter-cluster synchronization in MSTSP to internal attacks. As attack examples, an internal attacker may forge $t_$ex messages with incorrect receiving time values of the synchronization beacons from correspondent cluster reference nodes to desynchronize them. Two attackers may collaborate to wormwhole a cluster reference synchronization beacon to another cluster reference node far away. As another example, when becoming cluster reference node, an internal attacker may refuse to synchronize to its neighbor cluster

reference node even the latter is fastest. MSTSP provides two defense lines to ensure the inter-cluster synchronization against these malicious attacks:

- MSTSP takes the advantage of the topology redundancy of ad hoc networks to provide multiple synchronization paths via different bridge nodes to thwart the attacks to inter-cluster synchronization.
- The compromised or desynchronized cluster reference nodes can be detected by the bridge nodes by emitting alert messages when they detect that the clock difference is beyond the threshold. The synchronization procedure is then re-initiated.

In our approach, a collision of Hash chain elements may cause a security flaw that two or more nodes share their keys to secure their synchronization beacons. If one of them are malicious, it can impersonate others without being detected. Hereby we perform an analysis on the collision probability P_c: let A be the number of Hash chain elements in the element space, for m-bit Hash chains, $A = 2^m$; let N be the number of nodes in the network; let n be the length of the hash chains; let M be the total number of Hash chain elements, $M = N * n$. We thus have

$$\begin{aligned} P_c &= 1 - Prob\ (\text{no collision}) = 1 - \frac{C_A^M * M!}{A^M} \\ &= 1 - \frac{A!}{(A-M)!A^M} \\ &= 1 - \frac{A-1}{A}\frac{A-2}{A}\cdots\frac{A-(M-1)}{A} \\ &= 1 - \left(1-\frac{1}{A}\right)\left(1-\frac{2}{A}\right)\cdots\left(1-\frac{M-1}{A}\right) \\ &< 1 - \left(1-\frac{M-1}{A}\right)^{(M-1)} \sim 1 - \left(1-\frac{(M-1)^2}{A}\right) \\ &\sim \frac{M^2}{A}(M \ll A) \end{aligned}$$

Using 128-bit Hash chains in an ad hoc network of 1000 nodes using Hash chain containing 10^6 elements, we have $P_c < 10^{-28}$, which can be regarded as negligible.

5.4. Performance evaluation

We evaluate MSTSP by ns-2 [21] simulator. The clock parameters and the SSTSP parameters are the same as those in Section 4.8. The number of mobile nodes is 200 unless specifically stated. Each of them is randomly located in a 1000 m × 1000 m field with a transmission range of 250 m. All nodes move according to the random way-point model with maximum speed 5 m/s with the pause time 50 s. P_s is set to 0.6 unless specifically stated. We measure 3 metrics to evaluate the performance of MSTSP: maximum synchronization error, traffic overhead and synchronization latency.

Maximum synchronization error: IEEE 802.11 TSF is not scalable for multi-hop ad hoc networks. In the network of 200 nodes, the maximum synchronization error is already nearly 600 μs (Fig. 9). In contrast, MSTSP shows much better performance. As shown in Fig. 10 and Table 3, the maximum synchronization error of MSTSP is about 61 μs in the same scenario. We further vary the network size and simulate MSTSP in these configurations. The result is shown in Table 3. The average synchronization error of MSTSP is 30–50 μs. ASP is reported to achieve 100–200 μs in terms of synchronization error. The average synchronization error of the mechanism proposed in Rentel and Kunz [1] ranges from 50 to 200 μs depending on the network size and other parameters. To our knowledge, the accuracy of MSTSP is among the best of currently proposed solutions.

Traffic overhead: Traffic overhead is a crucial issue for time synchronization protocols in resource constrained environments. Some traditional synchronization mechanisms require each node to diffuse its local time values each synchronization period, resulting $o(n^2)$ traffic overhead. IEEE 802.11 TSF is very efficient in traffic cost, but its synchronization error in multi-hop ad hoc networks may uprise unboundedly. Fig. 11 shows the traffic overhead (number of synchronization beacons plus *t_ex* messages transmitted each BP) of MSTSP as P_s ranges from 0 to

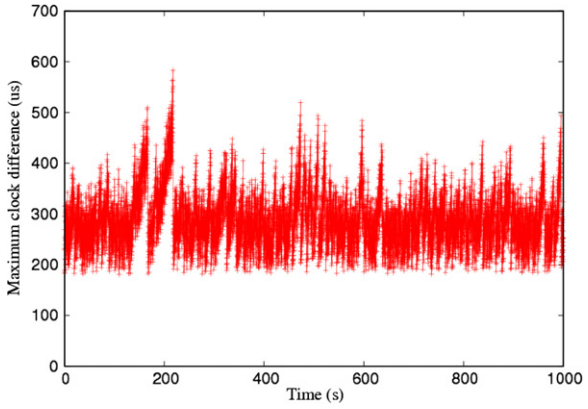

Fig. 9. Maximum clock difference, IEEE 802.11 TSF.

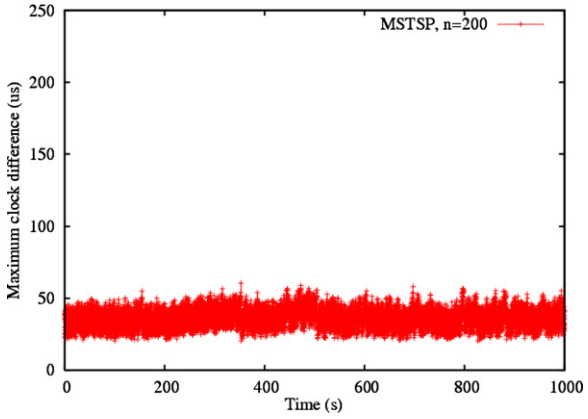

Fig. 10. Maximum clock difference, MSTSP.

Table 3
Synchronization error: MSTSP

Number of nodes	Maximum synchronization error (μs)	Average synchronization error (μs)
100	55	31
200	61	36
500	83	49

1. When $P_s = 0$, no synchronization beacon is relayed in *t*_ex messages, the traffic overhead equals to that of IEEE 802.11 TSF. When $P_s = 1$, all the bridge nodes relay the synchronization beacons in their *t*_ex messages. MSTSP is significantly more efficient than traditional approaches. Compared with IEEE 802.11 TSF ($P_s = 0$), MSTSP generates 2–5 times more overhead when P_s ranges from 0.3 to 0.7. We argue that the overhead of MSTSP is acceptable considering the improvement of performance in terms of synchronization precision and synchronization latency.

Synchronization latency: In order to simulate the synchronization latency of MSTSP, we attribute an initial clock offset between [−200 and 200 μs] to each node and measure the time between the beginning of the simulation and the time when the maximum synchronization error decreases under 100 μs. The result shows that with 100, 200 and 500 nodes in the network, the synchronization latency is under 10 μs. Besides, once the network is synchronized, MSTSP shows good stability without abrupt peaks in the maximum synchronization error as in IEEE 802.11 TSF.

Performance under attacks: Finally we study the performance of MSTSP in a hostile environment where 10% of nodes are compromised and forge the receiving time values in the *t*_ex messages they emit when becoming bridge nodes. Fig. 12 shows the maximum synchronization error of the rest 90% nodes with $P_s = 0.6$. We can see that with a proper P_s value, MSTSP can maintain the network synchronized even in hostile environments under malicious attacks. However, as discussed earlier in the paper, such attacks may cause detrimental effect on the performance of IEEE 802.11 TSF and other insecure synchronization protocols.

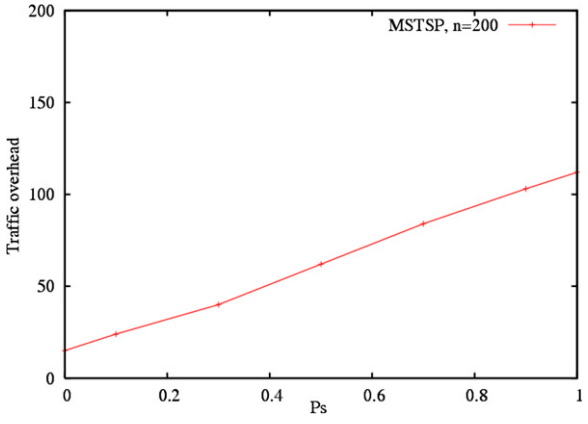

Fig. 11. Traffic overhead, MSTSP, 200 nodes.

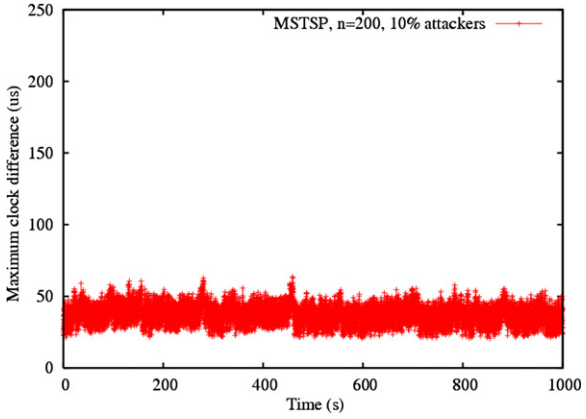

Fig. 12. Maximum synchronization error, MSTSP, 200 nodes, 10% attackers.

6. Discussion

It is interesting that our approach uses μTESLA to secure the synchronization process while μTESLA itself requires a loose synchronization. It is not contradictory in that the bootstrapping phase allow the new arriver to acquire initial synchronization and enables the application of one-way hash chains to secure the time synchronization procedure in the synchronization phase. Once the hash chain scheme is established and the synchronization is secured, we can continue to use the hash chain scheme based on the synchronized time to further maintain the time synchronization.

In this paper, we focus on detecting malicious attacks and preventing the network from being desynchronized by malicious nodes using erroneous time values. However, we do not provide recovery mechanisms when an attack is detected. We do not address how to eliminate the attackers either. We leave them for our future work.

In our approach, the key chain may be used up quickly. In this case, nodes need to re-issue new hash chains for μTESLA. To achieve this, nodes can broadcast the authenticated last element of the new hash chain to be used when the current chain is to be used up within a few time intervals. Better mechanisms include using 2-dimensional hash chain or interleaved hash chain to achieve seamless hash chain renewal. Jakobsson [7] addresses the hash chain renewal issue by proposing the infinite hash chain scheme.

7. Conclusion

In this paper, we address the security and the scalability problems of time synchronization protocols in ad hoc networks. For single-hop ad hoc networks, we propose SSTSP, a scalable and secure time synchronization procedure that significantly improves the performance of IEEE 802.11 TSF. We base SSTSP on one-way Hash chain, a

lightweight mechanism to ensure the authenticity and the integrity of the synchronization beacons. The "clock drift check" is proposed to counter replay/delay attacks. We then extend our efforts to the multi-hop case. We propose MSTSP, a secure and scalable time synchronization mechanism based on SSTSP for multi-hop ad hoc networks. In MSTSP, the multi-hop network is automatically divided into single-hop clusters. The secure intra-cluster synchronization is achieved by SSTSP and the secure inter-cluster synchronization is achieved by exchanging synchronization beacons among cluster reference nodes via bridge nodes. The proposed synchronization mechanisms are fully distributed without a global synchronization leader. We further perform analytical studies and simulations on the proposed approaches. The results show that SSTSP can synchronize single-hop networks with the maximum synchronization error under 20 μs and MSTSP 55–85 μs for multi-hop networks, which are, to the best of our knowledge, among the best results of currently proposed solutions for single-hop and multi-hop ad hoc networks. Meanwhile, our approaches can maintain the network synchronized even in hostile environments.

References

[1] C.H. Rentel, T. Kunz, Network Synchronization in Wireless Ad Hoc Networks, Technical Report SCE-04-08, Carleton University, Systems and Computer Engineering, July 2004.

[2] NTP Project, <http://www.eecis.udel.edu/ mills/ntp.html>.

[4] ANSI/IEEE Std 802.11, 1999 Edition.

[5] T. Lai, D. Zhou, Efficient and Scalable IEEE 802.11 Ad-Hoc-Mode Timing Synchronization Function, in: Proceedings of the IEEE 17th International Conference on Advanced Information Networking and Applications (AINA '03), Xi'an, China, March 2003.

[6] J. So, N.H. Vaidya, A Distributed SelfStabilizing Time Synchronization Protocol for Multi-hop Wireless Networks, Technical Report, UIUC, January 2004.

[7] M. Jakobsson, Fractal hash sequence representation and traversal, IEEE International Symposium on Information Theory (2002).

[8] H. Song, S. Zhu, G. Cao, Attack-Resilient Time Synchronization for Wireless Sensor Networks, in: 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS05), Washington, DC, November 2005.

[9] G. Khanna, A. Masood, C.N. Rotaru, Synchronization Attacks Against 802.11, Networks and Distributed Systems Symposium (NDSS) Workshop, San Diego, 2005.

[10] S. Ganeriwal, S. Capkun, C. Han, M. Srivastava, Secure Time Synchronization Service for Sensor Networks, WiSE 2005.

[11] J. Sheu, C. Chao, C. Sun, A Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks, ICDCS 2004.

[12] D. Zhou, T. Lai, A Compatible and Scalable Clock Synchronization Protocol in IEEE 802.11 Ad Hoc Networks, ICPP 2005.

[13] J. Elson, L. Girod, D. Estrin, Fine-Grained Network Time Synchronization using Reference Broadcasts, in: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.

[14] S. Ganeriwal, R. Kumar, M. Srivastava, Timing Sync. Protocol for Sensor Networks, ACM SenSys, Los Angeles, USA, 2003.

[15] Y. Hu, A. Perrig, D.B. Johnson, Ariadne: A Secure On-Demand Routing Protocol for Wireless Ad Hoc Networks, Technical Report TR01-383, Department of Computer Science, Rice University, December 2001.

[16] F. Stajano, R. Anderson, The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks, in: B. Christianson, B. Crispo, and M. Roe (Eds.), Security Protocols, 7th International Workshop, Springer Verlag, Berlin, Heidelberg, 1999.

[17] K. Römer, P. Blum, L. Meier, Time Synchronization and Calibration in Wireless Sensor Networks, Handbook of Sensor Networks: Algorithms and Architectures, John Wiley and Sons, 2005.

[18] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: Security Protocols for Sensor Networks, Mobile Computing and Networking, Rome, Italy, 2001.

[19] K. Sun, P. Ning, C. Wang, Secure and Resilient Clock Synchronization in Wireless Sensor Networks (Invited Paper), in: IEEE Journal on Selected Areas in Communications, vol. 24, No. 2, February 2006, pp. 395–408.

[21] The network simulator - ns2. <http://www.isi.edu/nsnam/ns/>.

Lin Chen is currently Ph.D. candidate in the department of Computer Science of ENST, Paris. He received his B.E. degree in Radio Engineering from Southeast University, China in 2002 and the Engineer Diploma from ENST, Paris in 2005. He also holds a M.S. degree of Networking from Paris University. His main research interests include security issues of wireless ad hoc networks and cooperation enforcement.

Jean Leneutre is an associate professor at the department of Computer Science and networks at ENST (French National School of Telecommunications), CNRS LTCI-UMR 5141 laboratory. He received his Ph.D. in Computer Science from ENST in 1998. His main research interests include security models and mechanisms for mobile ad hoc networks.