SWAN: A Secured Watchdog for Ad hoc Networks

Xiaoyun Xue†, Jean Leneutre†, Lin Chen† and Jalel Ben-Othman††

†Département INFRES - CNRS LTCI-UMR 5141, 46 rue Barrault, 75634 Paris Cedex 13 France
††Laboratoire Prism - CNRS UMR - 8144, Université de Versailles Saint-Quentin-en-Yvelines, 45 avenue des Etats-Unis, 78035 Versailles France

Summary
Due to its nature, ad hoc networks are much more vulnerable to various attacks than traditional wired networks. Many solutions have been proposed in recent researches for the security of ad hoc networks. The watchdog mechanism, based on a node supervising all its local neighbors, is one of the basic security mechanisms used by these solutions. It is able to detect both malicious attacks and selfish behaviors without significant overhead. However, it has a high storage requirement, and itself needs to be secured against spoofing attacks which may badly affect the reputation systems depending on it. In this paper, we propose a secured and efficient supervision mechanism based on the watchdog technique called SWAN. In order to avoid spoofing attacks, we combine the techniques SUCV (Statistically Unique and Cryptographically Verifiable) and TESLA (Timed Efficient Stream Loss-Tolerant Authentication) to provide a lightweight broadcast message authentication to watchdog. Moreover, we also propose an efficient storage method to reduce the storage overhead required by watchdog. Our analysis and simulations show that our approach is both lightweight and robust.

Key words:
Ad hoc networks, watchdog, broadcast message authentication, reputation system

1. Introduction

Mobile ad hoc networks are temporal and local area networks with no infrastructure, no a-priori organization but battery-based weak capacity nodes. The existing ad hoc security solutions, such as [1], [3], [4], [6], can be classified into three main categories: key management, secure routing and cooperation enforcement. The key management guarantees the identification and copes with all problems concerning keys; the secure routing uses the established keys to ensure the authentication, the confidentiality and the integrity in both the topology discovery and the data forwarding routing phases; and the cooperation enforcement fights selfish behaviors and encourages the cooperation between nodes.

Among all proposed solutions, the watchdog technique [6] is often used for the detection of selfish nodes and malicious attackers, and the results of those observations can feed some reputation systems, with which misbehaving nodes should be isolated and/or punished. CORE (a COllaborative REputation mechanism to enforce node cooperation) [9], CONFIDANT (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks) [10], TRP (Trust-based Routing Protocol) [11], etc..., are ad hoc routing protocols which adopt this kind of security system.

Watchdog uses the promiscuous mode which permits nodes to accept messages not sent to them. Compared to the other security techniques, the watchdog introduces neither additional traffic nor significant computation overhead, but it has a high level storage requirement for storing temporarily messages, and it needs to be secured against spoofing attacks since the latter can cause mistakes in reputation systems.

In this study, we propose a secured watchdog with an efficient supervision mechanism. First, in order to avoid spoofing attacks that may badly affect the reputation, we combine the techniques SUCV [12] and TESLA [13] to provide a lightweight Broadcast Message Authentication (BMA) to watchdog. A SUCV address ties naturally a Private Key (PK) and a node IDentifier (ID) together, thus no certificate server is needed to establish a PKI. Furthermore, by using IPv6, it is proved that a SUCV address is statistically unique, thus unspoofable. As for us, we use Hash chains to replace private keys in SUCV to achieve the authentication and the key management in SWAN. Therefore neither key pre-distribution nor central key server is needed by our scheme.

On the other hand, if we take a look at the majority of the secure proactive routing protocols [1], [14], [15], we can find that they use either Hash chain or TESLA to authenticate their routing messages/fields. Moreover, the scheme μTESLA proposed in [16] is an efficient BMA solution for sensor networks (also adaptable to ad hoc networks). We also use a TESLA-based authentication in watchdog since it is appropriate to treat a large number of messages.

We propose in addition an efficient storage scheme to reduce the storage requirement of watchdog, this without loss in the supervision capability. Instead of storing a whole message [6] or only the packet's identity and a hash on payload [9], we temporarily buffer packet's variable parts, a timestamp and a digest on packet's fixed parts.

Manuscript received June 5, 2006.
Manuscript revised June 25, 2006.

The organization of the rest of the paper is as follows: In section 2, we show the related works, and we specify our proposition in detail in section 3. In section 4, we make some discussions. We show how SWAN can be applied to an existing secure ad hoc routing protocol and some simulation results in section 5. Finally, we conclude the paper with section 6.

2. Related work

In this section, we mainly discuss the authentication mechanisms designed for the watchdogs that are used by existing reputation-based security protocols.

In CONFIDANT, a PKI similar to PGP is self-organized thus asymmetric cryptography can ensure the authentication of the routing control messages and ALARMs. Data packets are implicitly supposed to be sent on routes that are discovered and chosen for the purpose. However, this may not always be true, because in a mobile network a malicious node can pretend to be another node by spoofing the identity of the latter and then doing attacks. Therefore the attacks committed by the malicious node can decrease the reputation of the node spoofed. In CONFIDANT, no further mechanism is designed to address this issue.

In CORE, it is supposed that all the identities are unspoofable and unforgeable. Later, the authors have proposed a key management and message authentication scheme called IDHC. This scheme relies on an offline Key Distribution Center (KDC) server to provide an ID-based master ticket to each node. Afterwards each node should generate a series of authentication tickets based on its master ticket. The authentication tickets are used in a way similar to μTESLA, where each ticket is used during one time interval and the authentication is delayed to the next time interval. However, IDHC has a drawback in terms of computation overhead, since the generation of one authentication ticket is comparable to a RSA encryption, and the verification of a ticket is equivalent to a RSA signature verification.

Usual methods for BMA that we found in the literature are Hash chain [14], TESLA [15], μTESLA [16], etc... Among them, TESLA uses asymmetric keys to sign the first elements of Hash chains, and μTESLA uses symmetric keys to authenticate its Hash chains in sensor networks thanks to some pre-established trusts. Compared to them, the most important advantage of SWAN would be that it does not require a PKI or shared keys to authenticate the first elements of Hash chains. Furthermore, SWAN is carefully designed to be adaptive to reputation-based ad hoc secure routing protocols.

3. Our proposition

3.1 Assumptions

Neither additional module, nor a-priori key distribution or key server is required by the scheme, and nodes are not required to execute any asymmetric cryptography. Nevertheless, nodes should be able to compute hash according to a certain collision-resistant hash algorithm. Moreover, we suppose that the promiscuous mode is available to all nodes in the network, and IPv6 is in use.

To ensure a maximum correctness of watchdog, it is further assumed that all nodes in the network have a same transmission range and they all use an omni-directional antenna (so that all links could be bidirectional). CSMA/CA RTS-CTS may be used as the media access control protocol to reduce the "hidden terminal problem" [6].

Regarding the storage, nodes are expected to store at least one hash chain and to have a watchdog buffer in which unverified messages could be temporarily stored until the disclosure of their verification keys.

Since the source routing can facilitate the monitoring, we consider it as our underline routing algorithm. This choice is the same as in most of the existing reputation-based solutions, such as CONFIDANT and CORE. Thus adopting SWAN to them is supposed to be easy.

The following parameters are initialized and published in the network before SWAN is used:

(i) ***The lifetime of network T^{max}*** We assume that our ad hoc networks are temporal and local networks, and we are able to estimate the upper bound of the network lifetime T^{max}.

(ii) ***The time interval duration T^{ti}*** The network lifetime is split into time intervals of uniform duration T^{ti}.

(iii) ***The Network start time T_0*** It serves as a reference, to which all nodes synchronize their schedule of changing and disclosing keys.

The above parameters are not secrets and can be easily published in the network. Moreover, if a network is isolated, its IPv6 prefix may be chosen by itself. Then the parameters can be integrated into the IPv6 prefix:

$$\text{IPv6 prefix} = < (32\text{-bit})\ T_0, (16\text{-bit})\ T^{max}, (16\text{-bit})\ T^{ti} >$$

Finally, we assume a loose synchronization which guarantees an upper bound on the maximum synchronization error. In practice, the synchronization mechanism can be TSF (Time Synchronization Function) [18] or the solution proposed in [19] which better ensures the multi-hop synchronization in ad hoc networks.

3.2 SWAN scheme

3.2.1 Node initiation

The initiation of each node *I* should be done according to the following steps:

(i) calculation of the length of the Hash chain $n = [T^{max}/T^{ti}]+1$ ($[x/y]$ denotes y integer divides x).

(ii) generation of a Hash chain of n elements based on a random seed s_i: $h(s_i), h^2(s_i), ..., h^n(s_i)$.

(iii) setting of the temporary address of the node

$$AD_i = <\text{IPv6 prefix, hash-64}(h^n(s_i))>$$

where the last hash output length may be reduced to *63* or *62* or still less according to the length of the reserved bits in the IPv6 header.

The network time is then divided into n intervals $\tau_1, ..., \tau_n$ (only the last interval is incomplete). Any Hash chain element $h^{n-k+1}(s_i)$ ($1 \le k \le n$) will be used for the authentication of all messages sent or forwarded by node i during the time interval τ_k.

3.2.2 Message sending

Let M_{fix} be the fixed IP header fields (address of the initiator, address of the target, the packet identifier, etc.) and payload (if any) of a message *M,* and let M_{var} be the mutable fields of *M* (*TTL*, *hop_count*, etc.). In time interval τ_k, *M* sent by node *i* will be $<M, h_i^{n-k+1}(h(M_{fix}), M_{var})>$, where $h_i^{n-k+1}(A)$ indicates a HMAC output using the key $h^{n-k+1}(s_i)$ applied to a message *A*. Without loss of generality, *M* can either be a control packet or a data packet.

3.2.3 Message supervision[1]

Suppose that, in time interval τ_k, node I_i sends a packet *M* to node I_{i+1} and the latter should resend/forward it to node I_{i+2}. Then, the first message sent by I_i will be $<M, h_i^{n-k+1}(h(M_{fix}), M_{var})>$. I_i keeps the packet's identity, $h(M_{fix})$, and all mutable fields in its watchdog buffer.

Upon receiving the first message, I_{i+1} performs necessary modifications, and the modified packet M' is then sent to the node I_{i+2}: $<M', h_{i+1}^{n-j+1}(h(M'_{fix}), M'_{var})>$, where τ_j is the current time interval according to node $i+1$.

The packet M' sent by I_{i+1} will be observed by I_i. I_i identifies the packet thanks to its identity, checks M' mutable fields to see whether all modifications done by I_{i+1} respect well the routing protocol (whether a *TTL* is decreased by one, etc...), and finally it checks if $h(M_{fix}) = h(M'_{fix})$ for the integrity of the fixed fields of *M*.

If all above verifications are successful, so is the supervision. I_i can then increase its reputation for node I_{i+1}. Otherwise, a suspicious activity of node I_{i+1} is discovered and we have multiple choices:

(i) I_i can wait for the authentication phase to try to eventually identify I_{i+1} as a malicious node. This measure can prevent false negatives if nodes spoof when attacking.

(ii) I_i can directly decrease the reputation of node I_{i+1} without further verification. This measure can prevent false positives even though nodes spoof when attacking.

(iii) I_i can directly delete the entry from the watchdog buffer and leaves the reputation for I_{i+1} unchanged. This measure does not decrease reputations but only increases them when there are successful supervisions.

With the first choice, I_i stores $h_{i+1}^{n-j+1}(h(M'_{fix}),M'_{var})$ for future authentication.

3.2.4 Key disclosure and authentication

At each time interval $j+1$, any node *i* checks if it has sent any message during the previous time interval. If so, it discloses its key $h^{n-j+1}(s_i)$ by broadcasting a Key Disclosure message (KD) to its one-hop neighbors after the maximum synchronization error $<KD, h^{n-j+1}(s_i)>$.

Note that if there is any neighbor discovery process (that is the case of the most of the proactive and hybrid ad hoc routing protocols and also of some secure ad hoc routing protocols), by setting T^{ti} equals to the neighbor discovery interval, keys can be disclosed within neighbor discovery packets. This can reduce the message overhead.

Upon receiving a *KD* message, a receiver verifies at first whether the corresponding key is already validated. If so, it rejects the message. Otherwise, it verifies whether $AD_i = <IPv6\ prefix,\ hash\text{-}64(h^{j-1}(h^{n-j+1}(s_i)))>$.If the check fails, the key is rejected, and the node will still verify the other KD messages declaring the same key. Otherwise, the key is authenticated and stored, and the previous key discovered by node *i* is replaced.

3.2.5 Message authentication

Once a key $h^{n-j+1}(s_{i+1})$ is authenticated by node *i*, *i* checks in its watchdog buffer whether there is any message M' sent by node $i+1$ unauthenticated. If so, the validity of $h_{i+1}^{n-j+1}(h(M'_{fix}), M'_{var})$ is further checked.

If both the supervision and the authentication are successful, a good behavior is registered in favor of node $i+1$. Otherwise, if only the authentication succeeds, a bad

[1] Here we describe a supervision example using the mode supervision on route, with which only nodes involved in traffics observe what happens on routes. Actually SWAN can also support the mode supervision in neighborhood, where any node that is neighbor of both I_i and I_{i+1} can perform the supervision.

behavior will be attributed to node *i+1*. As a consequence, to have a good reputation, nodes have to behave correctly in both routing and key disclosing.

If the authentication fails, it is possible that *M'* is not sent by the node it claimed to be (spoofing attack), but we are not able to identify the attacker. Thus, to avoid false negatives, the result of the supervision may not be taken into account by the reputation system.

Note that with SWAN we mainly detect malicious behaviors. A selfish forwarding node can not be detected, since no (less) message is sent by it. To resolve the problem, we suggest to combine a solid neighbor lookup protocol with SWAN. Therefore a neighbor node which does not forward message can be considered selfish.

A whole SWAN process is shown in figure 1.

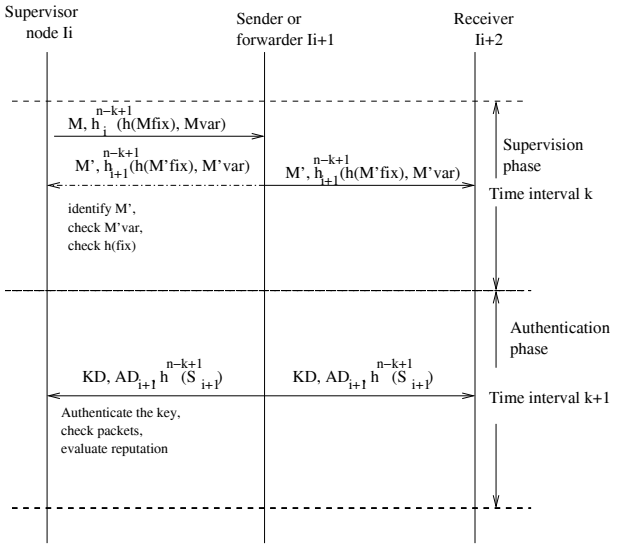


Fig. 1 A SWAN example

3.3 Capacity requirement analysis

3.3.1 Computation requirements

Table 1: Hashes required by SWAN

Operation	Number of hashes	Total hashed bits
Initiate a node	n + 2	64 + 128 * (n + 1)
Send a packet	2	128 + packet_length
Supervise a packet	1	≤ packet_length
Authenticate a key	j	128 * j
Authenticate a packet	1	128 + packet_var_length

Thanks to the temporary nature of ad hoc networks, in SWAN no asymmetric cryptography is required even in the node initiation phase. The only cryptographic operation, hashing, is a well-known lightweight cryptographic operation (The velocity of *160*-bit SHA-1 one-way hash function is *75*MB/s on a *33*MHz *486*SX).

Suppose a *128*-bit hash is used to generate the Hash chains and the seed length is also *128* bits, we can refer to table 1 to know the number of hashes and the total hashed bits required by each watchdog operation (note that in a real environment, the total hashed bits depends on the hash implementation).

Furthermore, by using *128* bits to store a before revealed key $h^{n-j+m}(s_i)$ ($2 \le m \le j$), it is possible to decrease the number of hashes required by the authentication of the key $h^{n-j+1}(s_i)$ to *m-1*, by checking whether $h^{n-j+m}(s_i) = h^{m-1}(h^{n-j+1}(s_i))$.

3.3.2 Storage requirements

We can refer to table 2 to know the number of bits to be stored by SWAN for each operation (under the same hypothesis as in the previous subsection):

Table 2: Memory required by SWAN

Operation	Length of total information to be stored (bits)
Hash Chain	$128 * ([T^{max}/T^{ti}] + 1)$
Authenticate and Supervise one packet	length(Timestamp + 2 * 128 +packet_identity)

To further reduce the memory space needed to store a *n*-element hash chain from *O(n)* to *O(log(n))*, the method proposed in [20] can be used. It selectively stores a logarithmic number of elements of a Hash chain, and the elements stored are modified over time, in such a way that we can easily find any hash value through several hashing operations.

3.3.3 Supplement overhead

In SWAN the only additional message is the key disclosure message. During the lifetime of a MANET, each node sends at most $[T^{max}/T^{ti}] + 1$ KD messages to its one-hop neighbors. Each KD message has less than *150* bits as length (IP header not included). And every traditional message has an additional overhead of *128* bits.

3.4 Security analysis

3.4.1 Reputation system security

We suppose that a benign node will always perform correct routing operations with its true identity and the correct authentication keys. Therefore, its reputations will be increased by the benign nodes observing it. Moreover, since SWAN guarantees that there is no spoofing attack, attackers are not able to decrease the reputations of benign nodes by spoofing their identities. Thus, we can ensure that at least the first-hand reputations for benign nodes will be correct.

We distinguish two cases of misbehaving nodes:

(i) If an attacker uses his true identity and his correct keys to attack, benign nodes can identify the attacker and decrease his reputation. Thus, there will be neither false negative nor false positive. However, in order to keep at least his bootstrapping reputation, an attacker would prefer to belong to the next category.

(ii) If an attacker does not use his true identity when attacking, or he uses wrong keys to make his attack messages unable to be authenticated, observing nodes will not be able to identify the attacker. Thus, the reputation of the attacker may not be decreased. However, since reputations of benign nodes do increase, a malicious node will still have its reputation inferior to those of the benign nodes.

Furthermore, since such a malicious node will have its reputation unchanged, we can consider that, if a node has existed in the network for a long period but still has its reputation unmodified, it is less trustworthy than a node having its reputation evaluated to a higher level or a new node.

Another solution is to periodically decrease all the reputations. Like this, even we can not correctly authenticate the attackers, attackers can not keep their bootstrapping reputations, and nodes must proactively participate into routing to have/maintain good reputations.

Considerations for the reputations of selfish nodes It is obvious that selfish nodes can not be directly detected by SWAN, because no (less) message will be routed by them.

Nevertheless, we distinguish selfish routing nodes from selfish forwarding nodes. Selfish routing nodes do not participate into the routing discovery and data forwarding, while selfish forwarding nodes only refuse to forward data packets. For the selfish forwarding nodes, even without authentication, we will be able to decrease their reputations because they refuse to forward data packets. As for the selfish routing nodes, they can at most keep their bootstrapping reputation values, thus they can not be trustworthy.

As conclusion, we believe that a reputation system using SWAN can produce relatively correct reputations between nodes.

3.4.2 Statistically unique address

Since ad hoc networks are usually local area networks with a limited number of nodes (for example, DSR requires that the MANET dimension is less than *16*), we believe that SWAN addresses are, like SUCV addresses, statistically unique if its hash algorithm is strong collision resistant (with a *n*-bit hash output, we need on average $2^{n/2}$ inputs to encounter an hash output collision).

Suppose that there are N nodes in a MANET, and the hash algorithm in use is a perfect p-bit hash algorithm. Then, the address collision probability in SWAN will be (let $A = 2^p$, $A >> N$, $N > 1$):

$$\begin{aligned}\text{Prob(collision)} &= 1 - \text{Prob(no collision)} = 1 - \frac{P_N^N C_A^N}{A^N} \\ &= 1 - \frac{A!}{(A-N)!A^N} = 1 - \frac{A-1}{A}\frac{A-2}{A}\cdots\frac{A-(N-1)}{A} \\ &= 1 - (1-\frac{1}{A})(1-\frac{2}{A})\cdots(1-\frac{N-1}{A}) \\ &< 1 - (1-\frac{N-1}{A})^{N-1} \sim 1 - (1-\frac{(N-1)^2}{A}) \sim \frac{N^2}{A}\end{aligned}$$

The collision probability increases with N and decreases with A. Since $A >> N$, the probability is small.

3.4.3 Unspoofable address and authentication

To successfully spoof an IP address, the following methods can be tried by attackers:

(i) ***Dictionary attack*** Also called ``brute-force attack''. An attacker can construct a database, also called a dictionary, which contains all possible pairs of $<seed, h^n(seed)>$. Thus, once a $h^n(s_i)$ is revealed, he can look up the corresponding s_i in the dictionary. However, the attack is difficult to realize since it is equivalent to break the one-way hash.

(ii) ***Replay*** It is somewhat true that without an accurate time information in packets, the replay attacks are possible. Nevertheless, we believe that replay in SWAN can not greatly affect the reputations of nodes. This is because, first, we do not take into consideration messages replayed in the same time interval of their first sending; second, messages replayed in a later time interval can not be successfully authenticated.

(iii) ***Finding a future key based on revealed ones*** Even though hash values will be revealed one by one by their owners, spoofing attacks can still be prevented because the hash operation is a one-way operation. The corresponding hash property is called ``weak collision resistance'', which means that given x, it is difficult to find a y that satisfies $h(y) = h(x)$. This attack is also equivalent to break the one-way hash.

3.4.4 Integrity

Since both mutable and fixed fields are protected by HMAC, the integrity of all the messages is ensured.

3.4.5 About bogus address

Due to the lack of countermeasure, in SWAN a malicious node can create a lot of bogus addresses in addition to its legitimate address. This attack permits attackers to bypass the reputation system by constantly appearing as a new node. Unfortunately, we can hardly prevent bogus addresses in absence of an online or off-line server.

Nevertheless, to complicate the generation of bogus addresses, we can use a binding of IP and MAC addresses. Since IP and MAC addresses are both unique and public,

nodes can not solely modify their IP addresses without changing their MAC addresses at the same time. Furthermore, to get each valid identity, an attacker should redo *n+1* hash operations, this also greatly complicates his task. Finally, since each of its addresses is only used temporarily, a node using bogus addresses can hardly get a high reputation thus it could not become highly trustworthy, let alone it uses these identities to attack.

Otherwise, we can also use the countermeasures against Sybil attacks presented in [5] to detect bogus addresses in SWAN.

3.5 Address renewal

Although we supposed that we can estimate the maximum network lifetime in a most pessimistic way, sometimes a MANET can exist longer than expected, or nodes could be too weak to support long hash chains, or the lifetime of the network is too long to be supported by one hash chain. In all these cases, node addresses must be renewed, but old reputations must not be lost. In this subsection, we propose two mechanisms that seamlessly link a new Hash chain to the old one without introducing additional messages.

3.5.1 Approach using overlapping Hash chain

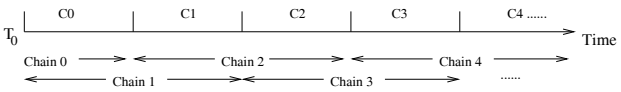


Fig. 2 Address renew using overlapping Hash chains

This solution consists of using two overlapping Hash chains, as shown in figure 2. During the node initiation phase, each node *i* picks two random seeds s_i^0, s_i^1 and generates two Hash chains: one of *n* elements based on s_i^0, the other of *2n* elements based on s_i^1. Node *i* then sets its temporary address in cycle *0* (in each cycle a node will use a different address): $AD_i^0 = < IPv6\ prefix,\ Hash\text{-}64(h^n(s_i^0))>$ and computes its address in cycle *1*: $AD_i^1 = <IPv6\ prefix,\ Hash\text{-}64(h^{2n}(s_i^1)) >$.

Then, in cycle *m*, at time interval τ_k, the format of a packet *M* sent by a node *i* will be the following:

$$<M,\ h_{m+1,i}^{2n-k+1}(h_{m,i}^{n-k+1}(h(M_{fix}),\ M_{var}))>$$

where $h_{m+1,i}^{2n-k+1}(A)$ denotes an HMAC applied to message *A* using the key $h_i^{2n-k+1}(s_i^{m+1})$, and $h_{m,i}^{n-k+1}(A)$ denotes an HMAC applied to message *A* using the key $h_i^{n-k+1}(s_i^m)$.

The message supervision is the same as that described in section 3.2.3 except that I_i stores $h_{m+1,i+1}^{2n-j+1}(h_{m,i+1}^{n-j+1}(h(M'_{fix}),M'_{var}))$ instead of $h_{i+1}^{n-j+1}(h(M'_{fix}),M'_{var})$.

The key disclosure and authentication in cycle *m* at time interval *j+1* is as follows: node *i* publishes both $h_i^{n-j+1}(s_i^m)$ and $h_i^{2n-j+1}(s_i^{m+1})$ in its KD message < KD, AD_i^m, AD_i^{m+1}, $h_i^{n-j+1}(s_i^m)$, $h_i^{2n-j+1}(s_i^{m+1})$ >. Then for the authentication of the keys, it is checked that $AD_i^m = < IPv6\ prefix,\ Hash\text{-}64(h^{n+j-1}(h^{n-j+1}(s_i^m))) >$, and $AD_i^{m+1} = < IPv6\ prefix,\ Hash\text{-}64(h^{j-1}(h^{2n-j+1}(s_i^{m+1}))) >$.

To authenticate messages, two HMAC operations have to be performed on $< h(M'_{fix}),M'_{var} >$ using the disclosed key pair $h^{n-j+1}(s_i^m)$ and $h_i^{2n-j+1}(s_i^{m+1})$. If the result is equal to the stored $h_{m+1,i+1}^{2n-j+1}(h_{m,i+1}^{n-j+1}(h(M'_{fix}),M'_{var}))$, the authentication is successful.

At time interval *n* of cycle *m*, to renew its Hash chain, node *i* should pick a new random seed s_i^{m+2}, generates a Hash chain of *2n* elements based on s_i^{m+2}, and then sets its temporary address in cycle *m+2* to $AD_i^{m+2} = < IPv6\ prefix,\ Hash\text{-}64(h^{2n}(s_i^{m+2})) >$.

This approach seamlessly links two Hash chains, and there is no additional overhead on the payload. However, compared to the original SWAN, each node has to store two Hash chains of *2n* elements instead of one Hash chain of *n* elements (except in the first cycle in which nodes store one chain of *n* elements and one chain of *2n* elements). In addition, one more HMAC should be computed when sending or authenticating a message.

3.5.2 Approach using Hash tree

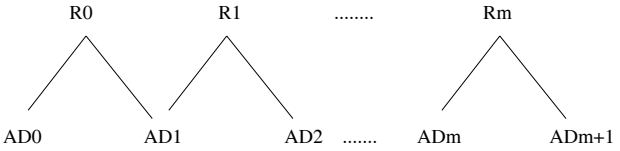


Fig. 3 Address renew using overlapping Hash tree

In this approach, a Hash tree is established as shown in figure 3. The leaves of the hash tree are Hash chains used in different cycles.

During the node initiation phase, each node *i* picks two random seeds s_i^0, s_i^1 and generates two Hash chains of *n* elements. Then, it sets its temporary address in cycle *0*: $AD_i^0 =< IPv6\ prefix,\ Hash\text{-}64(h^n(s_i^0)) >$ and its address in cycle *1*: $AD_i^1 =< IPv6\ prefix,\ Hash\text{-}64(h^n(s_i^1)) >$. The root of the Hash tree in cycle *0* is $R_i^0 = Hash\text{-}64(AD_i^0,AD_i^1)$.

In the *k*th interval of cycle *m*, the format of a packet *M* sent by a node *i* is: $<M,\ R_i^m,\ h_i^{n-k+1}(h(M_{fix}),\ M_{var},\ R_i^m) >$, where R_i^m is the root of the Hash tree in cycle *m*, $h_i^{n-k+1}(A)$ denotes an HMAC applied to message *A* using the key $h_i^{n-k+1}(s_i^m)$.

The message supervision process is the same as that described in section 3.2.3, since R_i^m is regarded as a fixed field. The message authentication in cycle *0* is also the same as that described in section 3.2.5.

In cycle *m* ($m \geq 1$), a Key Disclosure message will be < KD, AD_i^m, AD_i^{m-1}, R_i^{m-1}, $h_i^{n-j+1}(s_i^m)$>. To authenticate the key, three verifications are necessary:

(i) $AD_i^m =< IPv6\ prefix,\ Hash\text{-}64(h^{j-1}(h^{n-j+1}(s_i^m))) >$.

(ii) R_i^{m-1} is the same as the root published in cycle *m-1*.

(iii) $R_i^{m-1} = Hash\text{-}64(AD_i^{m-1}, AD_i^m)$.

In time interval n of cycle m, node i picks a new random seed s_i^{m+2}. It renews its Hash chain, generates a Hash chain of n elements based on s_i^{m+2}, and sets its temporary address in cycle $m+2$ to $AD_i^{m+2} =< IPv6\ prefix, Hash\text{-}64(h^n(s_i^{m+2})) >$. The new root of the Hash tree in cycle $m+1$ will be $R_i^{m+1} = Hash\text{-}64(AD_i^{m+1}, AD_i^{m+2})$.

Compared with the approach in section 3.5.1, this approach achieves its objective by adding more message overhead. But it has less calculation overhead.

4 Discussion

Choice of hash algorithm The longer is the hash output length, the heavier is SWAN but the better is its security. We estimate that a *64*-bit or longer hash algorithm is sufficient to reach the security requirements of SWAN.

However, recent progress in the cryptanalysis on MD5 and SHA-1 [21], [22] leads us to expect stronger hash algorithms.

Synchronization The synchronization is a common requirement in many solutions securing ad hoc networks. SWAN also needs a loose synchronization. An ideal synchronization mechanism for MANETs should be distributed and does not depend on a specialized hardware. One point worth mentioning is that the synchronization mechanism itself should be secured in order to provide secured "real" time.

New coming node and leaving node Leaving nodes do not take away any secret of network but only their personal secrets, so they leave without influencing the network security. Furthermore, a node out can return back to the network with a resynchronization, which will decide the number of keys to be skipped.

A new coming node should synchronize itself to the network by adopting the IPv6 prefix. It can use the value of T_0, T^{ti} and the current time T_c to calculate the number of the current time interval, then use the value of T^{ti} and T^{max} to compute its hash chain and identity.

Network dimension In order to have a weak address collision possibility, we suppose that SWAN is applied to the MANETs having a reasonable dimension. Unfortunately, SWAN will not be adaptable to very large MANETs as described in [23].

Duplicate address problem We discussed that SWAN addresses are statistically unique. But if ever we need to be certain of their uniqueness, the Neighbor Discovery Protocol (NDP) for IPv6 [2] can be used to resolve the duplicate address problem.

Immediate authentication If we need immediate authentication of routing control messages, the protocol ARIADNE [4] can be used. When sending a RREQ, the sender estimates the arrival time of the request to the destination node, and the intermediate nodes will use the hash values corresponding to that time to compute the HMAC outputs. Then, when the RREP message is being sending back, the intermediate nodes can be authenticated with their keys disclosed.

Influence of mobility In order to increase the authentication rate, KD messages can be sent to more nodes, such as 2 or 3 hop neighbors, if there is a strong mobility. This may also increase the reputation evolution velocity.

Participation to another network An address is valid only for the period of the current network. To participate to another network, nodes should be reinitiated.

Address renewal We insist that Hash chain renewal should rarely occur in SWAN. It is better to use one Hash chain in the whole network lifetime than dividing the entire network lifetime into some cycles and use one hash chain per cycle. This is because the address renewal introduces not only additional overhead and complexity, but also an important inconvenience due to the variation of IP addresses. Even though neighbor nodes can know the new IP addresses of each other, remote nodes are not easy to be informed.

Nodes do not change their identities within a cycle, so the problem appears only when a cycle finishes and the next one starts. We suppose that each cycle will be uniform and reasonably long.

When a proactive routing protocol is in use, there is periodic routing information exchanged within the whole network. Thus, the new addresses can be exchanged within the routing messages before the end of each cycle.

When a reactive routing protocol is in use, a source node can modify its RREQ message by adding its next address into it. Then, any node, once being a source node, can inform all the other nodes about its next address.

Even the next address of a destination node is unknown, a RREQ can still be sent to the old IP address (with a flag telling that it is an old address). Since the RREQ is broadcasted, the message will be received by the destination. Then, when a RREP is sent back, the new address can be joined.

Normally, intermediate nodes have necessary knowledge about their upstream and downstream nodes, because they are neighbors. This would be enough for the supervision. Moreover, new address information can also be accumulated in a RREQ (like in DSR we accumulate node identities) when the end of a cycle approaches. This will make all the new addresses on a route known to the whole route.

If there is an active data flow but the end of a cycle is reached, an additional message can be sent along and back the route to collect the new addresses on the route.

Finally, NDP [2] can also be modified to inform the variation of the addresses. That is, each node can send out a new address in *Neighbor Solicitation* message a little before the end of each cycle. If there is no new *Neighbor Solicitation* message during a timeout from the same node

(that means the address is not duplicated), other nodes can use the new address to replace the older address of the node in the next cycle. Note that nodes should also adjust their message sending times to avoid collision.

After all, we see that the address renewal process is quite complicate to manage, so it must only be used when it is strictly necessary.

SWAN applicability SWAN can operate with the source routing algorithm where a packet to be forwarded is perfectly predictable.

With the other routing algorithms, the receiving node of a packet forwarded may be decided (on-the-fly) by an intermediate node. Thus, a future packet is not entirely predictable, and the watchdog is not able to check all the fields of a packet. However, the rest the packets can still be supervised. Since SWAN is a generic security mechanism for watchdog independent of underline routing protocol, we believe that it can also be applied to other routing algorithms.

5. SWAN application and simulation

In this section we show a concrete example of the SWAN application and its simulation results. For the propose we have chosen our secure routing protocol TRP [11].

5.1 TRP overview

The Trust-based Routing Protocol (TRP) is a DSR-based secure ad hoc routing protocol. It combines the knowledge of misbehaving nodes with topology information, to help source nodes to choose the most reliable routes for their data sending.

In TRP, each node maintains a first-hand reputation for every other node it has encountered as neighbor. This reputation is computed based on the observed behaviors of the other node. During the discovery of a route, an intermediate node can inform the source node *S* of its first-hand reputations on its upstream and downstream nodes, by integrating them into the control messages (RREQ and RREP) of DSR. Thus *S* may receive a lot of routes and a series of reputations for each route. Then, based on the received reputations and its first-hand reputations, *S* can calculate an overall reputation for each route. Only a route obtained an acceptable overall reputation can be trusted and used to delivery data traffics.

TRP assumes a pair-wise key $K_{S,D}$ between the initiator and the target of a RREQ message. It uses a SRP-like (Secure Routing Protocol) [3] routing scheme and accumulates trust information during the propagation of RREQ. Trust information will be sent back to the initiator in a RREP message protected by a HMAC. The blackmail attacks are prevented thanks to our reputation exchange method.

Consider the following path *S, I_1, ..., I_i, ..., I_n, D*, the reputation value $C_{Ii, Ii-1}$ (the reputation of the node I_i on the precedent node I_{i-1}) will be added to a RREQ when the latter passes by I_i. For example, a RREQ that the destination node *D* returns would be:

$$D \text{ -> } S\text{: RREP, S, D, } Q_{seq}, Q_{id}, AD_{I1}, ..., AD_{I\_n}, C_{I2, I1}..., C_{D, In}, MAC_{K\,S,D}(\text{whole\_message})$$

More details can be found in [11].

5.2 TRP with SWAN

We discuss now how can SWAN provide the broadcast message authentication to TRP. In TRP, the fixed fields in RREQ and RREP are already protected by a MAC code. However, since only end nodes are able to verify the original MAC, a new MAC is required to provide authentication and integrity check to intermediate nodes during the supervision.

Suppose that at time interval τ_k, a node I_i rebroadcasts a RREQ which will be received by node I_{i+1}: I_i ->*: <RREQ, S, D, Q_{seq}, Q_{id}, AD_{I1}, ..., AD_{Ii}, $C_{I2,I1}$, ..., $C_{Ii,Ii-1}$, $MAC_{KS,D}$(RREQ, S, D, Q_{seq}, Q_{id}), $h_{Ii}^{n-k+1}(h(M_{fix}))$>, where M_{fix} = <*RREQ, S, D, Q_{seq}, Q_{id}, AD_{I1}, ..., AD_{Ii}, $C_{I2, I1}$, ..., $C_{Ii, Ii-1}$, $MAC_{KS,D}$(RREQ, S, D, Q_{seq}, Q_{id})*>. I_i stores < *RREQ,S,Q_{id}* > as packet's identity, *h(M_{fix},M_{var})* as the hash, and <AD_{I1},...,AD_{Ii}, $C_{I2,I1}$,...,$C_{Ii,Ii-1}$ > as the variable fields.

Upon receiving the packet, I_{i+1} should add to it its identity and its trust value on I_i before rebroadcasting it:

$$I_{i+1} \text{ ->*: RREQ, S, D, } Q_{seq}, Q_{id}, AD_{I1}, ..., AD_{Ii}, AD_{Ii+1}, C_{I2,I1}, ..., C_{Ii,Ii-1}, C_{Ii+1, Ii}, MAC_{KS,D}(\text{RREQ, S, D, } Q_{seq}, Q_{id}), h_{Ii+1}^{n-j+1}(h(M_{fix}), AD_{i+1}, C_{Ii+1, Ii})$$

where M'_{fix}=<*RREQ,S,D,Q_{seq},Q_{id}, $MAC_{KS,D}$(RREQ, S, D, Q_{seq}, Q_{id})*> and M'_{var}= <AD_{I_1}, ..., AD_{Ii}, AD_{i+1}, $C_{I2,I1}$,..., $C_{Ii+1,Ii}$>.

I_i observes the message and identifies the message. It further checks I_{i+1} and $C_{Ii+1,Ii}$ to see whether they are respectively a valid IP address and a valid trust value. Finally, it checks $h(M_{fix})$. For the future authentication, it stores AD_{Ii+1},$C_{Ii+1,Ii}$ and *j*.

During the next time interval, upon receiving the key $h^{n-j+1}(s_{Ii+1})$, I_i checks the validity of $h^{n-j+1}(s_{Ii+1})$ by computing *j* (or less) hashes: AD_{i+1}=<*IPv6 prefix, hash-64($h^{j-1}(h^{n-j+1}(s_{Ii+1}))$)*>. If success, I_i checks in addition whether $h_{Ii+1}^{n-j+1}(h(M'_{fix}),M'_{var})$ is valid.

The other types of messages, such as data, RREP and REER, do not change their contents during their forwarding. Therefore, for such a message *M*, M_{fix} equals to *M* and M_{var} equals to null. Their authentication can follow exactly the same process as described in section 3.2.

5.3 Simulation

We first implemented TRP under NS-2 [7], then SWAN is implemented on TRP (we call it TRPS later on). It calls the *crypto* library of OpenSSL [8] to realize the necessary cryptography operations. MD5 is chosen as our test hash algorithm.

The network that we simulated contains *25* nodes. The simulation time is *1,000* seconds, T^{ti} is set to *4* seconds and T_0 is set to *0* (thus each Hash chain contains *251* hash values). The simulation area is a *700m*700m* square. We use the way point mobility model with *5s* as pose time and *20m/sec* as the node maximum speed. Concerning the traffic, we consider the FTP application with at maximum *22* random CBR connections. Each connection sends *2* packets per second, and each packet has a *512*bit overload.

It is defined in our simulations that each watchdog buffer contains no more than *40* entries. When a new packet is to be buffered, the oldest packet in the buffer could be dropped if the buffer is already full. Each watchdog entry is in the following format: $<t, uid, AD_{sender}\text{-}64, h(M_{fix}), h_{keyed}(h(M_{fix}), M_{var}), supervised, authenticated, M'_{var}>$ where *supervised* and *authenticated* are two flags marking the states of the entry, *t* is the timestamp which registers the time when the entry is buffered. We do not save the prefix of the IP addresses.

At first, some performance simulations are carried out without attacker. We compared the total storage overhead of watchdog in TRP and in TRPS. The simulation results are shown in figure 4.

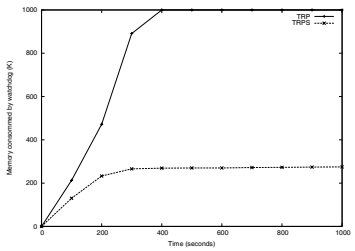


Fig. 4 Watchdog storage overhead in TRP and TRPS

We have also measured the end-to-end delay and the routing overhead of TRPS. We found that its average end-to-end delay is not varied compared to TRP, since nothing in SWAN can influence the traffic delay. As for the routing overhead, the additional KD messages represent about *19%* of the total number of network packets. But since the tested traffic has a low rate of *2*pkts/s, we believe that this percentage will drop with a higher data rate.

Then, our simulations were done in presence of 20% malicious attackers. Each attacker observes whether there is any data flow passing through its neighborhood. If so, he tries to spoof the addresses of its neighbors and send wrong packets. We show in Figure 5 that with TRPS, this attack can be avoided thus reputations of benign nodes will not be badly affected.

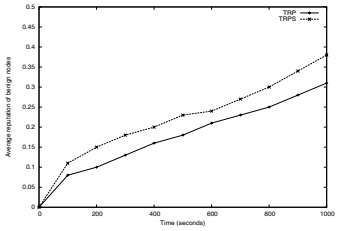


Fig. 5 Average reputation of benign nodes in TRP and TRPS

6. Conclusion

In this work, we proposed a secure watchdog for ad hoc networks named SWAN. It combines SUCV and TESLA to develop a watchdog with a lightweight broadcast message authentication mechanism. It can detect the spoofing attacks that may badly affect the reputation systems, and can reduce the storage overhead required by watchdog. It is also able to treat a large number of messages through a simple mechanism and be independent of any central server. Our analysis and simulations showed that SWAN is both lightweight and robust.

In the future, we plan to carry out some further simulations in presence of a random number of malicious attackers. We will also try to apply SWAN to other routing protocols besides DSR, more particularly to proactive protocols since they can offer the possibility to get rid of the overhead caused by key disclosure messages. Finally, we believe that the synchronization is also worth a deeper study, and formal security proves should be achieved.

References

[1] Yih-Chun Hu, David B. Johnson and Adrian Perrig, “SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad~Hoc Networks”, In Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02), USA, June, 2002.

[2] T. Narten, E. Nordmark and W. Simpson, “Neighbor Discovery for IP Version 6 (IPv6)”, RFC 2461, December, 1998.

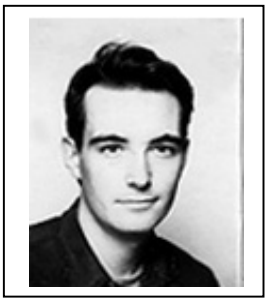
[3] Panagiotis Papadimitratos and Zygmunt J. Haas, “Secure Routing for Mobile Ad hoc Networks"," in Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, January, 2002.

[4] Y. Hu and A. Perrig and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", in Proceeding of The 8th ACM International Conference of Mobile Computing and Networking (ACM MobiCom 2002)", Atlanta, Georgia, USA, September, 2002.

[5] James Newsome, Elaine Shi and Dawn Song and Adrian Perrig, "The sybil attack in sensor networks: analysis & defenses", in Proceedings of the third international symposium on Information processing in sensor networks, Berkeley, California, USA, April, 2004.

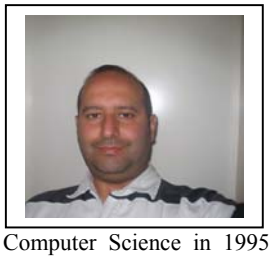
[6] Sergio Marti, T. J. Giuli, Kevin Lai and Mary Baker, "Mitigating routing misbehavior in mobile ad hoc networks", Mobile Computing and Networking (Mobicom 2000), pp.255-265, Boston, USA, 2000.
[7] The VINT project, "NS Notes and Documentation", UC Berkeley, LBL, USC/ISI, and Xerox PARC, May, 1998, work in progress.
[8] Mark J. Cox, Ralf S. Engelschall, Stephen Henson, Ben Laurie and al. The OpenSSL Project.
[9] Pietro Michiardi, "Cooperation enforcement and network security mechanisms for mobile ad hoc networks", PhD Thesis of ENST Eurécom, December, 2004.
[10] Sonja Buchegger and Jean-Yves Le Boudec, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks", in 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (Euromicro-PDP 2002), pp.403-410, Canary Islands, Spain, January, 2002.
[11] Xiaoyun Xue, Jean Leneutre and Jalel BenOthman, "A Trust-based Routing Prtocol for ad hoc networks", in Proceeding of Mobile and Wireless Communications Networks, pp.251-262, October, 2004.
[12] Gabriel Montenegro and Claude Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses", in Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS 2002), USA, February, 2002.
[13] Adrian Perrig, Ran Canetti and J.D.Tygar and Dawn Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", in IEEE Symposium on Security and Privacy, pp 56-73, Oakland, CA, May, 2000.
[14] Ralf Hauser, Tony Przygienda and Gene Tsudik, "Lowering security overhead in link state routing", Computer Networks (Amsterdam, Netherlands), Vol.31(8), pp.885--894, 1999.
[15] Steven Cheung, "An Efficient Message Authentication Scheme for Link State Routing", in Annual Computer Security Applications Conference (ACSAC 1997), pp.90-98, San Diego, CA, USA, December, 1997.
[16] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler and J. D. Tygar, "SPINS: security protocols for sensor netowrks", in Proceedings of the 7th annual international conference on Mobile computing and networking (ACM MOBICOM 2001), pp.189-199, Rome, Italy, 2001.
[17] David B. Johnson, David A. Maltz and Yih-Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", INTERNET-DRAFT draft-ietf-manet-dsr-10.txt, July, 2004, work in progress.
[18] LAN/MAN Standards Committee of the IEEE Computer Society, "Port-Based Network Access Control (802.1x Standard)", October, 2001.
[19] Carlos H. Rentel and Thomas Kunz, "Network Synchronization in Wireless Ad Hoc Networks", Technical Report SCE-04-08, Carleton University, Computer Engineering, July, 2004.
[20] Markus Jakobsson, Fractal hash sequence representation and traversal", in Proceeding of International Symposium on Information Theory (ISIT `02), pp.437-444, 2002.
[21] Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions", Eurocrypt, pp.19-35, 2005.
[22] Marc Stevens, "Fast Collision Attack on MD5", IACR ePrint archive Report 2006/104, 17 March 2006.
[23] K. Weniger and M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks", in Proceedings of European Wireless 2002, Florence, Italy, February, 2002.

Xiaoyun Xue is currently a PhD candidate in Department of Computer Science and Networking at ENST (French National School of Telecommunications), CNRS LTCI-UMR 5141 laboratory. She received her M.S. degree from University of Versailles in 2001 and her B.E. degree in Computer Science from Wuhan University of China in 1998. Her main research interest is the security issues in MANET.



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 PhD in Computer Science from ENST in 1998. His main research interests include security models and mechanisms for mobile ad hoc networks.

Lin Chen is currently a PhD candidate in Department of Computer Science and Networking at ENST (French National School of Telecommunications), CNRS LTCI-UMR 5141 laboratory. He received his Engineering degree from ENST Paris in 2005 and his B.E. degree in Telecommunications from Southeast University of China in 2002. He also holds a M.S. degree in Networking from Pierre et Marie Curie University of Paris. His research interest is the security issues in MANET.



Jalel Ben-Othman is an associate professor at the department of Computer Science at University of Versailles, PRiSM laboratory. He received his PhD in Computer Science from University of Versailles, France in 1998. He received the M.S. degree in Computer Science in 1995 from the University of Versailles, France. His main research interests include radio resource management, security and performance evaluation for wireless and mobile networks.