An Efficient Broadcast Authentication Scheme for Ad Hoc Routing Protocols

Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS 39762.
Email: ramkumar@cse.msstate.edu, Ph: 662 325 8435

***Abstract*— We introduce a novel broadcast authentication (BA) scheme especially well suited for nodes forming multi-hop ad hoc networks (MH-AHN). The proposed BA scheme is based on a recently proposed probabilistic key pre-distribution scheme (KPS), HARPS. Apart from a substantial performance improvements over existing BA schemes, the proposed technique offers many possible trade-offs (bandwidth-computation, and bandwidth-storage) for improving security.**

I. Introduction

A multi-hop ad hoc network (MH-AHN) can be seen as a collection of nodes, where each node has it unique "view" of the world around it. Distributed ad hoc routing protocols cater for dissemination of this topology information among nodes. The primary goal of all efficient routing protocols is to maximize information transmission while minimizing the necessitated use of resources like bandwidth / computational overheads.

Efficient solutions to the problem of routing in MH-AHNs can be challenging especially due to constraints on computational and bandwidth overheads. This problem is rendered even more complex under the presence of malicious nodes that could propagate misleading information. In order to prevent malicious nodes from modifying data reported by other nodes all data can be cryptographically authenticated. Such authentication may be meant for a *single* verifier or *multiple* verifiers. With single verifier authentication techniques, the source of the message has to append some authentication information for each possible verifier (based on a mutually shared secret). This is obviously inefficient for ad hoc networks where data originating from (or relayed by) a node may need to be verified by multiple nodes. Furthermore the source may not know *a priori*, the identities of the potential verifiers of the data. Thus multiple verifier authentication techniques or *broadcast authentication* (BA) is a very useful (and perhaps mandatory) security association for ad hoc networks.

The main contribution of this paper is an efficient BA scheme for MH-AHNs. The proposed scheme is based on a probabilistic key pre-distribution scheme (PKPS), HARPS (hashed random preloaded subsets) introduced by Ramkumar et al recently [1]. Similar to BA technique proposed by Canetti et al [3], BA using HARPS (or HARPS-BA) is achieved by appending many message authentication codes based on shared secrets (or MACs) in such a way that any verifier would be able to verify a *subset* of the appended MACs. Henceforth, we shall simply refer to the appended authentication data (or MACs) as the "signature" of the node[1]

The *strength* of (or the security offered by) any BA scheme is a measure of the difficulty an attacker faces in forging the signature of an arbitrary source, in order to fool arbitrary verifiers(s). The complexity of a scheme is a function of ① the computational and overheads required for evaluation of the signature, ② the bandwidth overhead (depending on the number of MACs that need to be appended and the size of each MAC), and ③ the number of MACs that need to be verified by each verifier. The *efficiency* of a broadcast authentication scheme is then a ratio of the strength to its complexity. We show that BA using HARPS is more efficient that BA techniques proposed in [3].

Any security solution involves overheads. The cost of any security solution, in general, has three contributors ① computation ② bandwidth and ③ storage. For mobile devices another major limiting factor (which in turn limits computational and bandwidth overheads) is ④ the efficiency of portable energy - or batteries.

The costs associated with storage is perhaps the least - even for mobile application scenarios (for instance flash based storage up to 8GB is already in the market). The cost of computation is steadily decreasing with time. Battery energy however may be a more expensive resource. Limited battery power may also influence the computational capabilities - after all, doing the same task faster takes more energy.

While bandwidth is not expensive for *wired* networks, for evolving scenarios with dense deployments of wireless devices, contention for channel may become a very serious issue unless bandwidth overheads are kept low. Furthermore in applications involving mobile wireless devices bandwidth directly influences rate of battery usage.

Thus efficient security solutions should offer possible trade-offs between resources. The proposed BA scheme offers efficient bandwidth-computation and bandwidth-storage trade-offs.

II. Broadcast Authentication with PKPS

A key pre-distribution scheme (KPS) consists of a KDC and N nodes with unique IDs. The KDC chooses a set of secrets $\mathbb{S}$. A node with ID A is provided with a set of secrets $\mathbb{S}_A$, which

[1]This "signature" does *not* cater for non-repudiation.

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

is a function of the KDC's secrets $\mathbb{S}$ and the ID of the node. As the secrets distributed to each node are *not* independent, an attacker who has gained access to secrets stored in many nodes can compromise all the KPS secrets. A n-secure KPS can resist compromise of all secrets from n nodes.

However probabilistic KPSs (PKPS), which provide probabilistic assurances, are more aptly described as (n, n_s, p)-secure. By exposing secrets from n nodes (say $\mathfrak{O} = \{O_1 \cdots O_n\}$), an attacker could discover *shared secrets* between arbitrary nodes (say A and B where $A, B \notin \mathfrak{O}$) with a *probability* p. Furthermore, an attacker may have to expose secrets from a substantially larger number $n_s >> n$ nodes (say $\mathfrak{O}' = \{O_1 \cdots O_{n_s}\}$) to expose *all* secrets in some node A (where $A \notin \mathfrak{O}'$) with probability p. In other words, PKPSs can withstand "eavesdropping" attacks (attacker needs to expose secrets shared between two nodes) with probability 1-p when n nodes have been compromised. However they can resist "synthesis" attacks (in which case the attacker has to expose every secret in some node) with probability 1-p even when a substantially larger number ($n_s >> n$) nodes have been compromised.

Leighton and Micali [4] (1993) proposed the first PKPS, LM-KPS, in which the secrets distributed to each node are repeatedly hashed versions of the secrets chosen by the KDC. The second PKPS was proposed by Dyer et al [5] (1995), which was based on random allocation of subsets of keys to each node (we shall refer to this scheme as RPS - random preloaded subsets). Earlier Gong et al [6], Mitchell et al [7] and Erdos et al [8] had considered KPSs with deterministic allocation of subsets. Dyer et al recognized that the complex allocation strategies in [7] which rendered such schemes impractical or naive allocation strategies [6] which rendered them inefficient, could be easily overcome with *random* allocation of subsets. More recently, Ramkumar et al proposed HARPS (HAshed Random Preloaded Subsets) [1], a generalization of LM-KPS [4] and RPS [5].

A. HARPS

In HARPS, defined by the set of parameters (P, k, L), where $\xi = k/P$, and functions $(F(), h())$, the KDC chooses and indexed set of P secrets $K_1 \cdots K_P$, and each node is provided with a *hashed* subset of $k = \xi P$ keys. The public random function $F()$ determines ① if a key corresponding to some index $1 \leq i \leq P$ is assigned to a node, and ② the "hash depth" of such a key. The hash depth refers to the number of times a key is hashed repeatedly (randomly and uniformly distributed between between 1 and L) using the cryptographic hash function $h()$. We shall represent by ${}^{j}K_i = h^j(K_i)$, the result of repeated hashing of K_i, j times, using the hash function $h()$.

Thus if $\{A_1 \cdots A_k\}$ are the indexes assigned to node A and $\{a_1 \cdots a_k\}$ their respective hash depths, the set of k preloaded secrets $\mathbb{S}_A$ assigned to node A are $\mathbb{S}_A = \{{}^{a_1}K_{A_1}, {}^{a_2}K_{A_2}, \ldots, {}^{a_{k_A}}K_{A_k}\}$. LM-KPS is a special case of HARPS with $P = k$, and RPS is a special case of HARPS with $L = 0$ (or keys are not hashed before pre-loading). Thus for RPS the secrets assigned to node A are $\mathbb{S}_A = \{K_{A_1}, K_{A_2}, \ldots, K_{A_k}\}$.

B. Broadcast Authentication with PKPSs

The basic idea used in broadcast authentication (BA) with preloaded subsets is very simple. The source of the broadcast appends the message with many key based MACs - one corresponding to each of the ξP (on an average) keys it possesses (or P keys if the source is the KDC). Any node will be able to verify the authenticity of the broadcast by verifying the MACs corresponding to the keys the verifier shares with the message source.

The "signature" of a node A, for a message M is thus

$$\mathfrak{S}_A(M) = [H_1 \parallel H_2 \parallel \cdots \parallel H_k]. \quad (1)$$

where

$$\begin{array}{lcll} H_i & = & h(A \parallel M \parallel K_{A_j}) & \text{(RPS)} \\ H_i & = & h(A \parallel M \parallel {}^{x_j}K_{A_j})),\ a_j \leq x_j \leq L & \text{(HARPS)} \end{array}$$

The primary reason for including[2] the ID of a node for calculation of the $H_i, 1 \leq i \leq k$ in each MAC is to ensure that the attacker cannot "pool" authentication data from different nodes for the same message M to forge the signature just by "requesting" such nodes to authenticate M. The only way for the attacker to forge messages is by ① actually tampering with devices and exposing buried secrets, or ② "guessing" the MACs.

The advantages of HARPS-BA over RPS-BA is due the flexibility of *choosing the hash depth* of the keys to be used for MACs. For example, if a node has the i^{th}key at a hash depth d (or the key ${}^{d}K_i$), the node can use any ${}^{x}K_i$ as the corresponding HMAC key, where $d \leq x \leq L$.

C. Analysis

Let us consider the scenario where the source is node A which appends k MACs. Any verifier will be able to verify, on an average, ξk of the k MACs. The MACs corresponding to any of the k keys is "safe" (cannot be forged by the attacker) if the following four conditions are satisfied:

① the verifier has a key corresponding to the index i, *and* the hash depth of the i^{th}key with the verifier (say d_v) is not greater than the hash depth of the i^{th}key used by the source (say d_s)

② the attacker coalition who have access to all secrets in n nodes (which does not include the source or the verifier) cannot discover the secret ${}^{d_s}K_i$ used by the source for computing the MAC

③ The attacker cannot "guess" the MAC corresponding to the i^{th}key. If each MAC is b bits long, the probability that the attacker cannot guess the MAC is $\alpha(b) = (1 - 2^{-b})$.

For now, we shall ignore condition ③. In other words, let us assume that the keys are long enough and the b is also large enough to render guessing of MAC bits infeasible. However,

[2]Timestamps and a random nonce could also be included for preventing replay attacks.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

in Section III we shall consider the case for small b, where we shall take condition ③ into consideration.

Let us represent the probability that conditions ① - ② are satisfied under the condition that an attacker has exposed all secrets from n nodes (which does *not* include the source or the verifier), or the probability that the i^{th}key is "safe," by $\epsilon(n)$. The probability p_F that the attacker can forge a message to impersonate an arbitrary source for purposes of fooling an arbitrary verifier is $p_F = (1-\epsilon)^P$.

If the source chooses the maximum hash depth L (or $d_s = L$) for every key then *any* verifier who has the i^{th}key will be able to verify the MAC. Or the scenario is identical to that of RPS-BA. If a source node chooses to use some other value of hash depth, say $d_s < L$, it may result in some nodes not being able to verify the MAC even though the verifying node *has* a key corresponding to that index (the verifying node may have key ${}^{d_v}K_i$ with $d_v > d_s$). The number of MACs a verifier can verify with HARPS-BA is between $k\xi/2$ (for the choice of $L_p = 0$) to $k\xi$ (for $L_p = L$, which is the same as RPS-BA). However, under the *optimal* choice of L_p, it is still less likely (as we shall see) that a coalition of attackers could forge the i^{th}MAC. The optimal choice which maximizes $\epsilon(n)$ (the probability that the i^{th}MAC key is safe) will depend on the parameters ξ, L and n.

The strategy is to choose the optimal depth (which could be periodically regulated by the KDC) *whenever possible*. Once a strategy is regulated, the source node does *not* have freedom to choose *any* possible hash depth. Also, given the strategy, any verifier (who knows all the hash depths of the keys that the source node possesses just from the ID of the source node using the public function $F()$) knows what hash depths have been (or should be) used by the source.

The source's strategy is to choose an optimal hash depth of L_p *whenever possible*. However, the source can choose depth L_p only for *some* keys - the source node would have roughly $\frac{kL_p}{L}$ keys with hash depth less than or equal to L_p, and $\frac{k(L-L_p)}{L}$ keys with hash depth greater than L_p. Or the source uses depth L_p with probability $\frac{L_p}{L}$, and uses some depth $j > L_p$ with probability $\frac{1}{L} \forall j > L_p$. If the hash depth used is L_p, the probability that a verifier (who has the i^{th}key) can verify the MAC is $\frac{L_p}{L}$. Similarly, if the chosen hash depth is $l > L - p$, the probability that a verifier can verify the MAC is $\frac{l}{L}$. The expression for the probability $\epsilon(n)$ that the i^{th}key is safe is provided in Table 1.

For RPS-BA on the other hand, the probability that both the source and destination share the i^{th}key is ξ^2. The probability that the attacker coalition does not have the key is $(1-\xi)^n$. Thus $\epsilon(n) = \xi^2(1-\xi)^n$.

Joint Verification by J Peers: In many scenarios it may be possible for $J > 1$ nodes to jointly verify the signature. Under this condition, any verifier will accept the signature as authentic only if $J-1$ other verifiers also verify the signature successfully. To fool J verifiers who "jointly" verify the signature, the i^{th}key is safe when the source and *any* of the J verifiers has the i^{th}key. For RPS, the probability that at least one of the J verifiers have the i^{th}key is $(1-(1-\xi)^J$. Thus $\epsilon(n, J) = \xi(1-(1-\xi)^J)(1-\xi)^n$. For HARPS-BA, the probability $\epsilon(n, J)$ that the i^{th}MAC key is safe is also reported in Table 1.

D. Performance Analysis

For RPS-BA, it can easily seen that the optimal choice of ξ that minimizes the number of MACs $k = \xi P$ that need to be appended for a target n and p_F, should minimize $p_F = (1-\xi^2(1-\xi)^n)^P \approx (1-\xi(1-\xi)^n)^k$, or maximize $\xi(1-\xi)^n$ - which is $\xi = 1/(n+1)$. For a deployment of RPS-BA optimized for some n, we cannot afford to efficiently cater for more than anticipated threat level, as ξ cannot be modified *post-deployment*.

Note that ξ is the factor which controls the fraction of appended MACs that verifiers can verify, and simultaneously the probability that an attacker can also forge the specific MAC. For HARPS-BA this factor - the fraction of MACs that can be verified (and forged by an attacker) is $\frac{\xi L_p}{L}$, which can be modified even after the deployment. Thus depending on the "threat level" an optimal value of L_p can be chosen. This is the main advantage of HARPS-BA over RPS-BA. Note that HARPS with $L_p = L$ is therefore identical to RPS (which can be easily verified by substituting $L_p = L$ in the equations in Table 1.

For example, $P = 1024, k = \xi P = 256$ RPS-BA can resist compromise of 4 nodes with probability $1 - 1.3 \times 10^{-9}$. However, for $n = 16$ the probability that an attacker can forge any signature is more than 0.5. For HARPS with $P = 1024, k = \xi P = 256$ and $L = 64$, by choosing $L_p = 16$, an attacker has to compromise more than 34 nodes to forge any signature with probability greater than 0.5 (and over 40 nodes if $L_p = 8$.

The second advantage of HARPS-BA over RPS-BA comes out of the significantly improved security under scenarios where signatures are jointly verified. For small L_p, each verifier verifies a lower number of MACs in HARPS when compared to RPS. Thus adding more and more verifiers helps HARPS substantially more than it helps RPS. For example with 5 joint verifiers, RPS can only tolerate 19 compromised nodes for $p_F \approx 0.5$ while HARPS can tolerate over 54 nodes. Even for 2 joint verifiers HARPS can "tolerate" 43 compromised nodes for $p_F = 0.5$.

Finally, for the ability to impersonate a node at will, for fooling every other node, the attacker has to perform a "synthesis" attack - or exposing all secrets stored in some node by exposing secrets from other nodes. For a successful synthesis attack with probability 0.5 and attacker has to expose all secrets from 488 nodes for HARPS ($P = 1024, k = 256, L = 64$) and only 21 nodes for RPS ($P = 1024, k = 256$). Thus the security of HARPS deteriorates very gracefully.

E. Computation-Bandwidth Trade offs

In many application scenarios, it is very crucial to limit the bandwidth overhead for BA - the number of bits required for the MACs. If we assume that each HMAC is b bits the

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

TABLE I

EXPRESSIONS FOR THE PROBABILITY THAT THE i^{th}KEY IS "SAFE" FOR HARPS-BA.

Sole verification by a peer

$$\epsilon(n) = \left(\frac{\xi L_p}{L}\right)^2 \left(1 - \frac{\xi L_p}{L}\right)^n + \sum_{l=L_p+1}^{L} \left\{ l \left(\frac{\xi}{L}\right)^2 \left(1 - \frac{\xi L_p}{L}\right)^n \right\}$$

Joint verification by J peers

$$\epsilon(n, J) = \frac{\xi L_p}{L} \left(1 - \left(1 - \frac{\xi L_p}{L}\right)^J\right) \left(1 - \frac{\xi L_p}{L}\right)^n + \sum_{l=L_p+1}^{L} \left\{ \frac{\xi}{L} \left(1 - \left(1 - \frac{\xi L_p}{L}\right)^J\right) \left(1 - \frac{\xi L_p}{L}\right)^n \right\}$$

probability that the attacker can guess the i^{th}HMAC is 2^{-b}. In this case, while the probability that the key used for the HMAC is safe is ϵ, the probability that the HMAC itself is safe is lower. Specifically, the probability ε that the HMAC is safe is $epsilon' = \epsilon(1 - 2^{-b})$. With BA schemes, we could reduce the number of bits for each HMAC and increase the number of MACs. The corresponding probability of forgery is $p'_F = (1 - \epsilon')^P > p_F$.

As $p'_F = (1-\epsilon')^P = (1-\epsilon(1-2^{-b}))^P \approx (1-\epsilon)^{P(1-2^{-b})}$, we can easily see that bandwidth-computation trade-offs are possible. In other words, instead of using large b for each HMAC, we can use say $b = 1$ bit for each HMAC and increase P (and consequently the $k = \xi P$, the average number of MACs that have to be evaluated) by a factor $1/(1-1/2) = 2$ (or more generally b bits for each HMAC and increase P by a factor $1/(1-2^{-b})$).

F. Verifier Complexity

The complexity of the verification process is perhaps more crucial for any BA scheme than the complexity of signing. The complexity for the source is evaluation of $k = \xi P$ MACs. The complexity of verification on the other hand is verification of $\xi k = \xi^2 P$ (the average number of *shared* keys) MACs for RPS. For HARPS, depending on the value of L_p the number MACs that can be verified is between $\xi k/2$ and ξk. It can be easily shown (see [3]) that the optimal choice of ξ for some n is $\xi^* \propto 1/(n+1)$. Under the optimal choice of $\xi \propto 1/(n+1)$ (for a desired n, p_F) $k = P\xi^* = n\mathrm{e}\log(1/\mathrm{p_F})$. Thus $k = \mathbb{O}(n)$ and $P = \mathbb{O}(n^2)$. Or the verification complexity is $\mathbb{O}(1)$. In other words the security of RPS-BA and HARPS-BA can be increased arbitrarily *without* increasing the verifier complexity.

III. BA FOR SECURING AHN ROUTING PROTOCOLS

The fact that the data advertised or broadcast by some node has been cryptographically validated does *not* imply that the data itself is valid. This would only be true under cases where the devices are completely *trusted*, and only such trusted devices are provided with access to secrets which could be used for authentication. For scenarios where fool-proof *tamper-resistance* is not possible, nodes with secrets that have turned malicious (by tampering with the software that controls the functioning of the device), could advertise misleading information that are cryptographically well authenticated. Similarly under scenarios where *read-proofing* of secrets is not possible, secrets extracted from trusted devices could be used by malicious devices to impersonate trusted devices.

For activities involving extensive mutual co-operation, for example routing in ad hoc networks, the "Byzantine generals" problem[3] [9] - [11] of discovering reliable routes (especially when resources are constrained) may be impossible when there are *multiple* colluding nodes.

It is for this reason that most secure routing protocols proposed in the literature [10]-[14] assume that "no two nodes collude together" of "no more than n nodes collude together." Practical deployments of AHNs would therefore need some assurances of tamper-resistance and read-proofing of secrets. Note that the need for read-proofing is more crucial than tamper-resistance. If an attacker is able to tamper with a device and render it malicious, the result is one malicious device. On the other hand, if an attacker is able to expose secrets from a device, he could construct *many* malicious devices which could impersonate the device. Furthermore the "morals" of such devices are only limited by the imagination of the attacker.

A. Attacker Strategy

Assuming that the sole motivation of an attacker is to disrupt routing protocols in MH-AHNs the attacker has many ways of achieving this:

A1: Broadcasting random packets with the intention of consuming bandwidth and consuming computational resources of other nodes. This is the easiest approach for any attacker as the attacker does not even require to tamper with devices or expose secrets from them. Keeping verifier complexity low is a good way to mitigate the effect of such attacks.

A2: By tampering with nodes and modifying their behavior (without compromising any secret). Such zombies are controlled by the attacker.

A3: By tampering with nodes and discovering all buried secrets. With this the attacker may be able to build multiple zombies corresponding to each node the attacker has exposed (all) secrets from. The "morality" of such nodes, obviously, is only constrained by the imagination of the attacker!

A4: Using secrets compromised from nodes to impersonate "healthy" nodes.

While the first three attacks are possible *irrespective of the KDS* used for BA, the third is the undesirable result due to the use of KPSs.

[3]Reaching a consensus among distributed units if some of them give misleading answers. The classical problem concerns generals plotting a coup, where some generals may be "moles."

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

1) Exploiting PKPS: While the attacker can impersonate the compromised nodes (nodes which have been physically compromised, and all secrets exposed) at will, for uncompromised nodes there is an associated probability of forgery. For example, a probability of forgery $p'_F = 1/1000$ implies that an attacker can impersonate a specific node for some specific message and fool one in a thousand nodes into accepting the signature. On the other hand $p_F = 1/1000$ (for which the attacker will have to compromise more nodes) implies that an attacker can impersonate a specific node for *any* message and fool one in a thousand nodes into accepting a signature. Furthermore if the attacker desires to impersonate any of 100 nodes, for $p_F = 1/1000$ the attacker can fool one in 10 nodes for forging any message.

From the point of view of an attacker who has tampered with and exposed all secrets from n nodes, there are 3 distinct "levels of confidence" of the attacker for purposes of impersonating nodes that the attacker has *not* physically compromised.

L1: The attacker can forge some messages by guessing some MAC bits (the attacker may not even know if such an attempt succeeded)

L2 The attacker knows that he can impersonate some nodes for purposes of fooling some verifiers for *any* message.

L3: the attacker knows that he could impersonate some nodes for purposes of fooling *any* verifier.

Recall that attacks on PKPSs could take the form of "eavesdropping" attacks (where attackers can discover shared secrets between nodes) or more expensive synthesis attacks (where every secret of a node is compromised by exposing secrets from other devices).

Level 2 of attacker confidence corresponds to successful eavesdropping attacks, and Level 3 corresponds to synthesis attacks. For Level 1 confidence the attacker does not even need to discover all shared keys (as he chooses to guess some MACs).

However any high impact attack (for example, if the attacker desires to create partitions in the network) will call for careful planning of the attack by the attacker and his ability to fool some very specific nodes in strategic locations. As such, the attacker faces many uncertainties in MH-AHNs as he would have very little control over the topology of the network. Additional uncertainties (for example if the attacker can fool only a fraction of the verifiers) may render such attacks more challenging. For launching useful attacks, the attacker may have to eliminate uncertainties regarding his ability to impersonate other nodes. In other words, the attacker may need to perform synthesis attacks. If the attacker discovers all secrets of a node, he can be sure that he can impersonate that node to *any* verifier. Synthesis attacks however are substantially more expensive for HARPS than RPS.

B. Increasing Storage

In devices where some proactive measures are taken to protect secrets, the secrets themselves are stored *encrypted* in non volatile memory. The secret used for encrypting all secrets is afforded a high level of protection. The encrypted secrets, or the NVM, does not need any protection. The practical implication of this is that the *number* of secrets to be stored is not an issue. For instance secrets could be stored in pluggable flash memory (like SD cards). Given that SD cards of 8GB capacity are already in the market, the storage complexity associated with secrets is of no concern.

A simple strategy involving increased storage could substantially increase the complexity of attacks aimed at achieving levels 2 and 3 of confidence. The strategy is to increase both P and k by a factor $\beta >> 1$. In other words each node *stores* βk keys on an average instead of k.

However the nodes *use* only k of the βk keys. The specific indexes of the keys used by the source could be dictated by the message itself. In other words, each signature involves the same computational complexity[4], verifier complexity and bandwidth overheads.

Note that while such a strategy will not reduce the per-message forgeability probability (which would be based only on the number of keys used), it would make well planned attacks more difficult. In other words level 1 confidence for this scheme is the same as the earlier scheme. However, for achieving level 2 or level 3 confidence, the complexity for the attacker faces is equivalent to that of compromising a $(P\beta, k\beta)$ scheme!

For instance, if $\beta = 100$ for $k = 256$ the number of secret each node needs to store is 25600. If each secret is 128 bits (16 bytes), it implies a storage of 400 KB for each node - an insignificant part of the storage that could be available for any mobile device. Increasing storage 100 fold increases the resistance of synthesis attacks 100 fold. The per-message-forgeability probability is still the same as the case of $k = 256$ as only 256 keys are used for each signature. Reducing per-message forgeability probability would require increasing the number of keys used for each signature.

What increasing storage does is to keep the attacker confidence between level 1 and level 2 in practice. As even eavesdropping attacks could become significantly expensive for large choices[5] of β, an attacker will not even have level 2 confidence - that he could impersonate a node for fooling a specific node *for every message* (as the choice of the keys would be dictated by the message itself). Finally, this approach also makes it possible to increase k (though undesirable from the point of view of resource consumption) and decrease L_p, under increasing threat levels.

IV. Conclusions

We introduced a novel broadcast authentication scheme and analyzed many of its desirable features that make it well suited for securing ad hoc routing protocols. The proposed scheme is a significant improvement on a previously proposed scheme also based on PKPSs. While the use of KPSs for

[4]Apart from the additional complexity required for evaluation of the one way function which determines the indexes of k of βk secrets that source should employ for the particular message.

[5]In practice even $\beta = 1000$ should be easily feasible.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

security associations will place some reliance on infeasibility of attackers to expose secrets from a large number of devices, it is important to note that the need for such assurances is not because we *need* to use KPSs. Such assurances are needed in any case for AHNs irrespective of the key distribution scheme used.

With improvements in technology the costs associated with each resource is expected to reduce. However, for MH-AHNs, it is still likely that bandwidth overheads and battery consumption will remain the most expensive. One intuitively appealing property of key pre-distribution schemes (KPS), especially probabilistic KPSs (PKPS) is their ability to perform many trade-offs depending on the cost of resources. In particular, they can make use of inexpensive resources like storage to reduce reliance on more expensive resources like bandwidth.

Furthermore with improvements in technology to provide assurances of read-proofing, the overheads (especially bandwidth) associated for BA with KPSs will *reduce* further. However with increasing computational capabilities, the overheads associated with public key schemes will only *increase*. Note that the security of KPSs rely on the our ability to prevent an attacker from exposing secrets from a large number of devices. On the other hand, for public key schemes, however well technology caters for protection of private keys, the security of such schemes are based on the computational infeasibility of determining private keys.

REFERENCES

[1] M. Ramkumar, N. Memon "An Efficient Key Pre-distribution Scheme for MANET Security," IEEE Journal on Selected Areas of Communication, March 2005.

[2] N. Alon, "Probabilistic Methods in External Finite Set Theory," in *Extremal Problems for Finite Sets,*, pp 39-57, 1991.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," INFOCOMM'99, 1999.

[4] T. Leighton, S. Micali, "Secret-key Agreement without Public-Key Cryptography,"*Advances in Cryptology* - CRYPTO 1993, pp 456-479, 1994.

[5] M. Dyer, T. Fenner, A. Frieze and A. Thomason, "On Key Storage in Secure Networks," *Journal of Cryptology,* **8**, 189–200, 1995.

[6] L. Gong, D.J. Wheeler, "A Matrix Key Distribution Scheme," *Journal of Cryptology*, **2**(2), pp 51-59, 1990.

[7] C.J. Mitchell, F.C. Piper, "Key Storage in Secure Networks," *Discrete Applied Mathematics,* **21** pp 215–228, 1995.

[8] P. Erdos, P. Frankl, Z. Furedi, "Families of Finite Sets in which no Set is Covered by the union of 2 Others," *Journal of Combinatorial Theory, Series A,* **33**, pp 158–166, 1982.

[9] L. Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, Vol. **4**, (3), pp 382–401, July 1982.

[10] M. Burmester, T. Van Le, M. Weir, "Tracing Byzantine Faults in Ad Hoc Networks," Proceedings of Communication, Network, and Information Security (CNIS), NY, Dec 2003.

[11] B. Awerbuch, D. Holmer, C. Nita-Rotaru, H. Rubens. "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures," ACM Workshop on Wireless Security (WiSe-02), September 2002.

[12] P Ning, K Sun ,"How to Misuse AODV: A Case Study of Insider Attacks against Mobile Ad-Hoc Routing Protocols," Proceedings of the 2003 IEEE Workshop on Infomation Assurance, United States Military Academy, West Point,NY, June 2003.

[13] X. Du, Y. Wang, J. Ge, Y. Wang, "A Method for Security Enhancemens in AODV Protocol," Proceedings of the 17th International Conference on Advanced Information Networking and Applications, AINA'03, Xian, China.

[14] T Wan, E kranakis P.C. Van Oorschot,"Securing the Destination Sequenced Distance Vector Routing Protocol," Proceedings of the 6th International Conference on Information and Computer Security, Malaga, Spain, October 2004.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.