Proceedings of the 2006 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY

Securing Ad Hoc Networks With "Asymmetric" Probabilistic Key Predistribution Schemes

Mahalingam Ramkumar

Abstract— **We present two "asymmetric" probabilistic key predistribution schemes to cater for mutual authentication and broadcast authentication respectively. The schemes however employ only symmetric cryptographic primitives - the asymmetry is due to the use of different secrets for encryption / authentication and decryption / verification, which are however related through a one-way function. Both schemes try to take advantage of an abundant and inexpensive resource, storage, to improve their security. While both schemes can have a wide range of applications, we limit ourselves to their utility in securing multi-hop ad hoc networks.**

I. Introduction

A multi-hop ad hoc network (MH-AHN) can be seen as an autonomous collection of nodes, where each node has it unique "view" of the world around it. Distributed ad hoc routing protocols thus cater for efficient exchange of such topology information, to determine optimal routes between any two nodes. While efficient solutions to the problem of routing in MH-AHNs can be challenging due to constraints on computational and bandwidth overheads that can be tolerated by resource constrained, battery operated devices, it is rendered even more complex under the presence of malicious nodes that propagate misleading information.

A first step towards securing routing protocols is thus to cryptographically authenticate such information, as it is well known that this *Byzantine generals* problem[1] is more tractable if source authentication is possible [1].

Cryptographic authentication techniques rely on the ability to establish different types of security associations (SA) like mutual authentication, and broadcast authentication, facilitated by key distribution schemes. Obviously, while cryptographic authentication can render the Byzantine generals problem more tractable, they will still call for substantial overheads, unless most devices taking part in such co-operative activities are sufficiently trustworthy.

In this paper we propose two novel probabilistic key predistribution schemes (PKPS), 1) asymmetric random preloaded subsets (A-RPS) for mutual authentication and 2) MSBA (multi source broadcast authentication). Both schemes involve simple variations of techniques involving *random* allocation of subsets, first pioneered by Dyer et al [2] in 1995, and an elegant extension proposed by Canetti et al [3] to permit the use of Dyer's subset allocation scheme for broadcast authentication by multiple sources. While both schemes may have a wide range of applications, we focus primarily on their utility for securing interactions between devices forming co-operative multi-hop ad hoc networks.

Mahalingam Ramkumar, Department of Computer Science and Engineering, Mississippi State University, Mississippi.

[1] 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."

A. The Cost of Providing Assurances

The overheads associated with establishment of SAs take the form of computation, bandwidth and storage. However the costs associated with different types of overheads are not the same. For wireless mobile devices while bandwidth and computational resources may be expensive, *storage* is the least constrained, and also exhibits the fastest Moore's law growth rate with no impending sign of saturation. Even for mobile hand held devices like PDAs and mobile phones with add on flash memory capabilities, storage of the order of GBs are already available. The two schemes presented in this paper try to exploit this abundant and inexpensive resource to improve security.

Section 2 provides an overview of key distribution schemes, with more focus on probabilistic key predistribution schemes (PKPS). The two schemes are discussed in Sections 3 and 4.

In Section 5 of this paper we argue that security solutions for any application scenario calls for preventive and corrective measures, and thus efficient security solutions should strive to minimize the costs associated with such measures. We argue why, the proposed approach synergistically reduces the cost of preventive and corrective security measures for securing ad hoc networks. Conclusions are offered in Section 6.

II. Key Predistribution Schemes

Key predistribution schemes (KPS) consist of an off-line key distribution center (KDC) who chooses a set of P secrets, and N nodes with unique IDs. Each node is provided with a set of k secrets. The set of secrets provided to each node is a function of the P secrets chosen by the KDC and

1-4244-0130-5/06/$20.00 ©2006 IEEE

the *unique* ID of the node. The main advantage of KPSs, their ability to cater for ad hoc authentication (without ongoing involvement of a trusted authority) without using asymmetric cryptographic primitives however, comes at price. KPS SAs are susceptible to collusions of devices (or an attacker who has compromised and can pool together secrets from many devices).

Typically, the size of such attacker's pools that can be tolerated is proportional to the number of keys k that need to be stored in each device. Thus the security of KPSs (or the pool-size of compromised nodes that can be tolerated) can be increased to any extent by increasing storage.

In any scenario where multiple secrets are used, it is common practice (which dates back to at least 1978 [4]) to use a master secret to encrypt all other secrets. Thus the encrypted secrets can be stored even in unprotected storage locations[2]. Thus the *number* of secrets that devices need to store is not any longer a crucial issue. After all, even if device is allocated 1 million 64-bit secrets, the storage required is mere 8 MB. With the rate at which storage capabilities are growing, in a few years even a GB of storage for any hand-held communication / computing device may be practically "free."

Unfortunately, for most KPSs the number of secrets allocated to each device also determines the computational complexity and, in some instances like broadcast authentication, the bandwidth overheads. So the question now is, *how can this abundant and inexpensive resource, storage, be used for improving security - without increasing computational and bandwidth overheads*?

A. Probabilistic Key Predistribution

KPSs based on the concept of allocation of a subset of keys to every node, from a larger pool of keys, have been extensively studied in the literature. The earliest of such approaches involving simple allocation strategies, was by Gong and Wheeler [5] (1990). Later Mitchell and Piper [6] (1995), considered more complex allocation strategies, influenced by the work of Erdos et al [7]. Dyer et al [2] (1995), perhaps influenced by Alon's [8] (1991) works on probabilistic methods in finite sets, investigated probabilistic allocation strategies, or strategies involving random preloaded subsets (RPS).

Canetti et al [3] (1999) proposed various broadcast authentication schemes based on the RPS scheme by Dyer et al. The more recent re-emergence of subset allocation schemes in the literature is due to Escheneur et al [9] (2002), in the context of securing sensor networks, following which this area has seen substantial activity.

RPS can be defined by two parameters P and k. The KDC chooses an indexed set of secrets $\mathbb{S} = \{K_1, K_2, \ldots, K_P\}$. Every node in the network, with a unique ID, is assigned a subset k of the P secrets. The specific k keys allocated to any node may be determined by a random one way function seeded by the node ID. Thus for node A, such a function $F()$ can be used to determine the indexes $F(A) = \{A_1, A_2, \ldots, A_K\}$ assigned to node A. Corresponding to the indexes assigned, node A is provided with k secrets $\mathbb{S}_A = \{K_{A_1}, K_{A_2}, \ldots, K_{A_k}\}$.

Any two nodes will share, on an average $\bar{m} = k^2/P$ indexes. For mutual authentication of any two nodes, say A and B, the nodes can independently discover the $m \approx \bar{m}$ shared indexes by evaluating $F(A) \cap F(B)$. The corresponding m shared secrets are hashed together to evaluate K_{AB}, the secret used for mutual authentication of A and B.

However, an attacker who has exposed all secrets from n nodes (say nodes with IDs $M_1 \cdots M_n$) can determine K_{AB} (or all m elementary secrets A and B share) if $F(A) \cap F(B) \in \{F(M_1) \cup F(M_2) \cup \cdots \cup F(M_n)\}$. It can be easily shown that the probability $p(n)$ that the attacker (who has pooled secrets from n nodes) can discover K_{AB} is

$$p(n) = (1 - \xi(1-\xi)^n)^k, \text{ where } \xi = k/P, \tag{1}$$

and the optimal choice of ξ that minimizes k is $\xi = 1/(n+1)$. Such a scheme which is n-secure with probability $1-p$, is referred to as (n,p)-secure. For an (n,p)-secure scheme, an attacker who has pooled secrets from n devices can determine one in $1/p$ SAs like K_{AB}.

A.1 Source Authentication

In the "basic scheme" by Canetti et al [3], where the source is the KDC, to authenticate a message M the source appends P key based message authentication codes (MAC) - one corresponding to each of the P secrets in $\mathbb{S}$. Any verifier can verify $k = \xi P$ of the P appended MACs. An attacker who has exposed secrets from n nodes can impersonate the source (KDC) for purposes of fooling A if $F(A) \in \{F(M_1) \cup F(M_2) \cup \cdots \cup F(M_n)\}$. The probability $p_F(n)$ that the attacker will be successful in discovering all secrets that A has (in which case the attacker can calculate all ξP MACs that A can verify) is

$$p_F = (1 - \xi(1-\xi)^n)^P \Longrightarrow \begin{cases} \xi^* = 1/(n+1) \\ P \approx \mathrm{e}(\mathrm{n}+1)\log(1/\mathrm{p_F}) \\ k \approx \mathrm{e}\log(1/\mathrm{p_F}) \end{cases} \tag{2}$$

and minimizing the number of appended MACs P for some n and p_F once again entails choice of $\xi = 1/(n+1)$,

In the same paper, Canetti et al also proposed an elegant extension of the basic scheme to cater for broadcast by external sources who are not provided with any of the KPS secrets. For example, if such an external entity W desires to broadcast, W obtains P values $\mathfrak{S}^W = \{K_i^W = h(K_i \parallel W)\}, 1 \le i \le P$. Now all broadcasts by W are

[2]Obviously, under these conditions, compromise of all secrets from a device amounts to compromising the single master secret - which is afforded a high level of protection.

authenticated with P MACs using the secrets $\mathfrak{S}^W$. As in the case of broadcasts by the KDC, any verifier can verify ξP MACs. Thus the performance - in terms of p_F and n is identical as the case where the KDC was the source.

In scenarios where bandwidth is a expensive (for example, in wireless ad hoc networks), appending many large MACs for authenticating messages may not be practical. Canetti et al also considered possible bandwidth-computation trade-offs. For example, while each of P secrets could be 128 bits long and the MACs that are evaluated are also 128 bits long, we could append only the LSB of each MAC. This however opens up a new line of attack for the attackers, in the form of guessing the MAC bit without having to compromise the corresponding secret.

In other words, we now have some probability p_F that attackers can determine all secrets by exposing secrets, and a probability p'_F that they can *guess* some bits without having to expose secrets. Cannetti et al argued that by choosing $(4P, 4k)$ scheme with one bit MAC is at least as secure as a (P, k) scheme with large[3] MACs.

III. Mutual Authentication Using Asymmetric RPS

A-RPS, for mutual authentication, is in fact very similar to the *source* authentication scheme proposed by Canetti et al to cater for external sources. For the realization of an (n, p)-secure A-RPS scheme, the KDC chooses an indexed set of $P = k(n + 1)$ secrets $\mathbb{S} = \{K_1, K_, \ldots K_P\}$, where $k = \mathrm{e}\log(1/\mathrm{p})$. Each node is provided with two sets of secrets - one set of k *decryption* secrets, and a second set of P *encryption* secrets.

More specifically, a one way function $f()$ seeded by the ID of a node and an index, generates a random integer between 1 and $n+1$. For a node A the function $a_i = f(i, A)$ is evaluated for $1 \leq i \leq k$. Corresponding to each a_i the index $A_i = (i-1)n + a_i, 1 \leq a_i \leq n+1, 1 \leq i \leq k$ is assigned to node A. Thus node A is assigned k decryption secrets $\mathbb{S}_A = \{K_{A_1}, K_{A_2}, \ldots, K_{A_k}\}$. In addition, node A is also assigned P encryption secrets

$$\mathfrak{S}_A = \{K_i^A = h(K_i \parallel A)\}, 1 \leq i \leq k \tag{3}$$

Similarly, node B receives k secrets $\mathbb{S}_B = \{K_{B_i}\}$, where $B_i = f(i, B), 1 \leq i \leq m$, and P encryption secrets $\mathfrak{S}^B = \{K_i^B = h(K_i \parallel B)\}, 1 \leq i \leq P$. The shared secret between A and B, K_{AB} is now evaluated as $K_{AB} = h(S_1 \parallel S_2 \parallel \cdots \parallel S_k)$ where $S_i = h(K_{B_i} \parallel A)$, $1 \leq i \leq k$.

More specifically, node A evaluates $B_i = f(i, B)$, k times to determine the indexes of the k secrets it needs to use from $\mathfrak{S}^A$ for evaluating K_{AB}. Note that $k = \log(1/p)\mathrm{e}$, does *not* depend on n. Only the storage for the $P = k(n+1)$ authentication secrets depends on n. For instance if $p = 10^{-20}$, $k \approx 128$. With 64 MB of storage, assuming 64 bit or 8-byte secrets, we can afford $P = 2^{23}$ - or $n = 2^{16}$. In other words, an attacker who has exposed all secrets from over 65 thousand devices can still determine only one in 10^{20} security associations like K_{AB}.

Obviously with more storage - for example with 512 MB of storage - A-RPS can resist compromise of all secrets from over half a million devices! Note that increasing the number of secrets does not in any way increase the computational complexity - either for evaluating the public function $f()$ or the number of symmetric cipher operations (m). Thus A-RPS can be made as secure as we desire by just increasing the storage complexity. It is also important to note that network size is only limited by the number of bits assigned to represent the ID of any node (each node requires a unique ID).

We refer to this scheme as *asymmetric* RPS due to the use of separate secrets for encryption and decryption (A-RPS does *not* use any asymmetric cryptographic primitives). The encryption secrets of node A do not provide any information about the secrets $\mathbb{S}$. Thus compromise of A's encryption secrets affects only node A. However the decryption secrets are subsets of the k KDC secrets $\mathbb{S}$. Thus (n, p)-security for A-RPS implies that an attacker who has compromised *all decryption secrets* from n nodes can compromise decryption secrets of nodes (not belonging to the set of n compromised nodes) with a probability p.

The statement that "compromise of A's encryption secrets only affects A" however holds only for application scenarios that do *not* call for extensive mutual co-operation. For applications like ad hoc networks, even the encryption secrets of A have to be protected *from* A. Obviously, we do not wish the owner of device A to have the ability to validate *any* (possibly misleading) information he / she chooses to propagate!

Nevertheless, the fact that the number of decryption secrets k are very small compared to the number of encryption secrets (P) can be useful in many application scenarios. In many scenarios nodes A and B may have vastly different capabilities. For instance node A may be a PDA used to query a tiny wireless sensor B. As another example B may be a infra-red / blue-tooth remote control device operating a more capable set top box A. In such scenarios it is enough for node B to have access to just the m decryption keys.

Arguably, even with very ineffective assurances of protecting secrets, just the fact that an attacker may have to gather collusions of *hundreds of thousands* of devices, can for all practical purposes, render the problem of "susceptibility to collusions" of KPSs moot. Furthermore, the improvement in security is realized only by using storage. Note that the computational complexity for this scheme is indeed trivial - and does not depend on n.

[3]Large enough to render the option of guessing impractical.

IV. Multi Source Broadcast Authentication

One of the application areas where the capability of multi source BA (MSBA) is very important is in multi-hop ad hoc networks, where information emanating from a node may need to be verified by many other nodes. Furthermore, as the source may not know a priori, the identities of potential verifiers, authentication of messages individually to each verifier is not feasible.

Note that the A-RPS scheme described earlier, is actually very similar to the BA scheme by Canetti et al [3]. Thus the BA scheme can also be made as secure as we desire by increasing P. Unfortunately increasing P implies increasing the number of appended MACs - as the source appends P MACs. Obviously, for wireless applications, even with one bit MACs, large P will not be acceptable.

Let us therefore assume that the BA scheme is restricted to a bandwidth of 512 bits for the appended MACs - say 512 1-bit MACs. Given this limitation, the question now is, *what is the best we can do*?

A. Impersonation Attacks: Guessing vs Key Compromise

Though Canetti et al [3] weigh the two risks, 1) the risk of attacker exposing the secret used for computing MACs and 2) the risk involved with the attacker's ability to guess MACs *equally* (the authors argue that $p_F = p'_F \approx 2^{-20}$ may be "reasonable" in many application scenarios), obviously (as they themselves point out) success by *guessing* is not as favorable for the attacker as the ability to actually compromise the keys.

In situations where the attacker has to resort to guessing a few MACs, the attacker has no way of knowing *a priori* if the impersonation attempt is going to succeed, and possibly, in some situations, may *never* come to know. Furthermore while the attacker may succeed in impersonating A for fooling B for some message, success is not guaranteed for the next message - for which the attacker will have to guess all over again.

Especially in multi-hop ad hoc network scenarios where the attacker may need to carefully orchestrate attacks where a series of packets (with authenticated misleading information) may need to be propagated, guessing - even if the attacker has to guess just one bit in every attempt, cramps the attacker's ability to launch high impact attacks. On the other hand, in scenarios where the attacker knows all secrets corresponding to the indexes B can verify, the attacker can *consistently* impersonate *any* node for the purpose of fooling B.

Thus we can afford a higher probability of impersonation by guessing (per message), but will prefer a substantially lower probability of success of attacks through compromise of secrets. Let us assume, for the sake of illustration, that while we can tolerate success by guessing with probability 1/16, but require that the probability that an attacker can actually reveal all secrets (that can be verified by some verifier) be of the order of 2^{-30}.

Note that the probability that a coalition of n attackers (or an attacker who has exposed secrets from n nodes) do not have access to the key corresponding to the i^{th}MAC (or the key $K_i, 1 \leq i \leq P$) is $(1-\xi)^n$. The probability that a verifier can verify the i^{th}MAC is ξ. Thus the probability that the i^{th}MAC is safe from key compromise attacks is

$$\varepsilon(n) = \xi(1-\xi)^n. \tag{4}$$

Permitting the attacker a guessing probability of 1/16, is the same as requiring that at least 4-bits of the P appended MACs are safe (or 4 MACs are safe if each MAC is 1-bit long). The probability that the attacker will need to guess less than w MACs is

$$p(n,w) = \sum_{u=0}^{w-1} \binom{P}{u} \varepsilon^u (1-\varepsilon)^{P-u} \tag{5}$$

What we desire then, is $p(n,4) \approx 2^{-30}$. The best that we can do (given the bandwidth limit of 512 bits) is $n = 6$, by choosing $P = 512$ and $k \approx 512/7 \approx 73$ - in which case $p(6,4) = 2^{-30.7}$.

B. Making Use of Storage

As storage is not an issue, we can afford to increase P and k. At first sight this strategy does not seem very useful as we need to limit the number of appended MACs. However, as we shall argue, it is indeed possible to make use of storage to improve the resilience of the scheme.

Consider a scenario where instead of employing (P,k) A-RPS we choose $(P' = \alpha P, k' = \alpha k)$ A-RPS, and $\alpha >> 1$. However, the source *appends* only P of the αP possible MACs. For instance, the specific P of the αP indexes chosen for any message, can be dictated by a one-way function of the message to be authenticated. As in (P,k) A-RPS, any verifier (with αk verification secrets) can verify k of the P appended MACs (on an average).

As long as $w\alpha$ of the $k\alpha$ secrets of any node are safe (with a high probability), then the attacker has to resort to guessing (on an average) w bits for any MAC. The probability that a particular verification secret of a node (say B) is safe from an attacker who has exposed all αk verification secrets of n nodes (that does not include B) is $p_s = (1-\xi)^n$. Thus the probability that less than $w\alpha$ secrets of B are safe is

$$\epsilon = \sum_{i=0}^{w\alpha-1} \binom{k\alpha}{w\alpha} p_s^i (1-p_s)^{k\alpha-i},\ p_s = (1-\xi)^n \tag{6}$$

In other words, as long as ϵ is sufficiently low, the attacker has to resort to guessing w bits on an average for forging any message.

For $P = 512$, $w = 4$, $\epsilon < 10^{-12}$ (or $\epsilon \approx 2^{-40}$), if we choose $\xi = 1/(n+1)$, the maximum n that can be tolerated for different values of α are shown in the table below:

α	10	20	50	100	1000
n	21	25	32	36	43

For example, for $\alpha = 100$, each node stores $P' = 512 \times 100$ encryption secrets and $P'/37 \approx 1383$ decryption secrets, requiring a mere 420 kilobytes of storage. While the case for which $\alpha = 1$ could reasonable resist only compromise 6 nodes, by increasing storage we have achieved a six fold improvement ($n = 36$). While the pay-off reduces exponentially, storage is inexpensive.

C. MSBA Using Asymmetric HARPS

While Dyer was perhaps the first to suggest probabilistic subset allocation strategies, the first key pre-distribution scheme with probabilistic guarantees was suggested by Leighton and Micali [10]. More recently, Ramkumar et al proposed hashed random preloaded subsets (HARPS) [11], a generalization of Dyer's scheme and the scheme in [10]. HARPS [11] is defined by three parameters, P, k, L.

As in RPS, the KDC chooses a set of P indexed secrets $\mathbb{S} = \{K_1, K_2, \ldots, K_P\}$, and a public one way function $F()$ determines the indexes $F(A) = \{A_1, A_2, \ldots, A_K\}$ assigned to node A. However, HARPS uses a second public function $a_{A_j} = f(A, j), 1 \leq j \leq k$, where a_{A_j} are random integers between 1 and L. Now node A is assigned k secrets

$$\mathbb{S}_A = \{K_{A_1}^{a_{A_1}}, K_{A_2}^{a_{A_2}}, \ldots, K_{A_k}^{aA_k}\}, \;\; K_{A_j}^{a_{A_j}} = h^{a_{A_j}}(K_{A_j}), \quad (7)$$

where $K_i^d = h^d(K_i)$ represents the result of successive hashing of K_i, d times - or K_i at "hash depth" d.

Similar to the "asymmetric" extensions of RPS, an asymmetric realization of HARPS (A-HARPS) is also possible. In this case, the KDC chooses P secrets and provides every node with $k = \xi P$ verification secrets at random hash depths. The authentication secrets provided to each node are however at a fixed hash depth, say d. In other words, node A receives secrets

$$\mathfrak{S}_A = \{h(K_i^d \parallel A)\}, 1 \leq i \leq P. \quad (8)$$

Let the verification secrets of node B be $\mathbb{S}_B = \{K_{B_i}^{b_{B_i}}\}$, $1 \leq i \leq k$. While B has k verification secrets, unlike the case of RPS-A B cannot verify MACs corresponding to all indexes B_i. It can verify only MACs corresponding to indexes B_i for which $b_{B_i} \leq d$. Thus on an average, B can verify only a fraction $\gamma = \frac{d}{L}$ of the k indexes (or a total of $k\gamma$ MACs). The expression for the probability that any MAC is safe, $\epsilon(n)$, is similar to that of A-RPS, with a small difference - instead of ξ for RPS we have $\xi\gamma$ for A-HARPS. In other words, instead of a $(P = 512, k = 32)$ A-RPS we can use $(P = 512, k = 128, L = 64)$ A-HARPS with $\gamma = 0.25$ (or $d = 16$) or $(P = 512, k = 64, L = 64)$ A-HARPS with $d = 32$ ($\gamma = 0.5$) - all of which have identical performance.

In heterogenous deployments, information originating from different nodes may have different importance. Depending on their importance, some may use 512 bit signatures while some may use 1024 (say broadcast by cluster heads in routing protocols employing such techniques) or even 2048 bits (for example, broadcasts by the KDC). With an $(\alpha P, \alpha k)$ A-RPS, designed for efficient operation for $n = 25$ (by choosing $P/k = n + 1 = 26$), while we can always use $2P$ or $4P$ MACs, we cannot do so *efficiently*. For the same scheme, by increasing P to 2048 we can only resist compromise of 60 nodes. Note that to realize a *proportional* increase in n with P, we need choose P/k optimally (as $P/k \propto n$).

With A-HARPS however, nodes that use $P = 512$ may be provided with authentication secrets with depth $d = L$ (or $\gamma = 1$). Nodes that use 2048-bit "signatures" on the other hand can be provided with $d = L/4$ (or $\gamma = 1/4$). Thus when the signature size is increased by a factor 4 we can increase n by the same factor - *with the same verification secrets.*

V. Preventive and Corrective Measures

Any security solution includes preventive measures which ensure that security breaches will not occur in the first place, and corrective measures to cater for the failure of preventive measures - for example to unambiguously identity perpetrators responsible for such breaches with the intent of revoking their privileges.

In the widely deployed client-server (C-S) based applications today where cryptographic authentication finds extensive use, in the final analysis, it is still the individual human beings on whom the *onus of trust* is placed. For instance, end-users (or clients) authenticate themselves to a server using a shared secret (a pass-phrase). As anybody with access to the secrets of the user can impersonate the user, the end-users are trusted not to disclose their secrets.

This trust is not generally misplaced as every user has the motivation, and responsibility, to protect their secrets. Furthermore, compromise of private information of a user does not result in a great inconvenience for other users. It only affects the irresponsible users (who do not protect their secrets), and indirectly, the users who trust the irresponsible user. The responsibility is enforced by the fact that abuse of privileges by a client, or by some one who compromises the secret the client was entrusted with, can be attributed to the client, in order to take corrective actions like revoking the privileges of the irresponsible client.

In the same manner, if every node in an ad hoc network "accepts responsibility" for the information it provides (say by appending a signature), at first sight *it would appear* that it is possible to attribute the origin of misleading / malicious information to their perpetrators, who may face the risk of being evicted from the deployment (which indirectly forces them to behave in a responsible manner). Unfortunately, in application scenarios calling for a high degree of interdependence, the risk of an attacker "getting

caught in the act," and especially being held *accountable* for it, can be extremely low. This is even more true under scenarios where resource constraints do not permit *unlimited* overheads.

In a scenario where a malicious node B sends some misleading information, the first requirement is for some node A to *detect* that there is indeed something inconsistent in the information provided by B. Even this step, calls for redundancies (for example node A may receive the same information from a few other nodes).

Even under circumstances where it may be possible for a node to detect inconsistencies in the information provided by B (say based on information provided by other nodes C and D), it may well be possible that B is indeed providing the correct information while the information provided by both C and D are misleading. The second, and more difficult requirement is the ability to unambiguously *attribute* the source of misleading information - which obviously requires more redundancies (compared to case of mere detection of inconsistencies).

Note that there are parallels between well known paradigms between error detection / error correction codes [12] and detection / attribution of inconsistencies in routing information. Just as error correction will require more redundancies than error detection, unambiguous *attribution* of errors will call a more redundancies - or more overheads in routing protocols. Furthermore, just as overheads for error correction can increase substantially with increasing number of errors that have to be corrected, under scenarios involving many *colluding* nodes, the overheads required can be prohibitively high.

Even in the event that node A has "strong reasons to believe" that the information supplied by B is indeed misleading, possibly the best that it can do is to make a note of this fact, and in the future drop all packets sent by B. Theoretically, while a collection of nodes (who are convinced that node B has malicious intents) could jointly take steps to "revoke" B from the network, such steps are easily susceptible to simple denial of service attacks, where nodes could propagate misleading information exclusively for creating unnecessary traffic. Thus the risk of attacker getting caught in the act, and especially being held *accountable* for the it, may indeed be very small in ad hoc networks.

In other words, we cannot simply afford to rely on the users who control devices "to act responsibly" just because they "accept responsibility" by cryptographically authenticating the information they advertise. Thus in practical deployments of devices forming ad hoc networks, the devices themselves have to be *trusted*.

A. Trustworthy Devices

Trustworthy devices should cater for some extent of tamper-resistance and read-proofing. Tamper-resistance is necessary to ensure that the software controlling the devices, cannot be changed. Read proofing is necessary to ensure that secrets protected by trustworthy device, which will be used for authentication of the device, cannot be exposed. Note that without assurance that the software executed by the device cannot be modified, even without exposing secrets an attacker can force the device to cryptographically authenticate any (mis)"information." Similarly, by exposing secrets of a device, an attacker can create any number of devices (perhaps deployed at different locations) that can propagate any authenticated "information" the attacker wishes to disseminate.

Any trusted computer defines a clear "boundary of trust." *Enforcing* the trust boundary involves providing assurances of components inside the boundary, through countermeasures that shield the components from attacks that can 1) reveal secrets or 2) modify the code / data stored, inside the boundary.

One undesirable side-effect of shielding components from intrusions (using physical *active* and *passive* shields) is that they render the problem of *heat dissipation* difficult. Active shields may consist of fine wire meshes [13], [14], which will block microprobes and picoprobes [15] - or focused ion beams that can be used to expose secrets by tapping electrical signals passing through the buses or to set / reset bits in specific memory locations. Passive shields are required to block electromagnetic radiations emanating from within the chip (which can be used to reveal information about secrets) and for preventing external radiations from injecting faults [16]. Solutions that cater for effective shielding *and* effective heat dissipation tend to be expensive [13].

B. Synergistic Preventive and Corrective Measures

However if we limit the scope of such trustworthy components (included in devices taking part in co-operative activities) to purely symmetric cryptographic primitives, and keep the complexity of such components at very low levels, we can eliminate the need for proactive measures for heat dissipation - thereby eliminating constraints on techniques available for effective shielding. Thus reducing the complexity of trustworthy components can simultaneously render them inexpensive *and* trustworthy.

Improvements in capabilities of mobile computing devices indicate that the capability to perform asymmetric cryptography is not beyond the capability of any conceivable device that could take part in a multi-hop wireless ad hoc network. However, it is very much desirable that the trustworthy components employed by such devices, that perform sensitive operations (like evaluation of security associations) still have a very low complexity.

Preventive measures call for improving trustworthiness of devices taking part in the network. Corrective measures have the ultimate intent of revoking privileges of perpetrators responsible for providing misleading information. While carrying out corrective measures to the fullest extent

may not be practical, *defensive corrective measures*, like 1) detecting inconsistencies, leading to such packets being dropped, and 2) attributing the source of inconsistency, however without irrepudiable proof, leading to scenarios where the victim may drop all packets from the suspected perpetrator in the future, may be possible.

Restricting ad hoc networks to symmetric cryptographic primitives can also render the overheads, in terms of computation and bandwidth for 1) redundancies that will facilitate corrective measures, and 2) cryptographic security associations, low. For wireless networks, bandwidth is perhaps the most expensive of resources, especially in dense deployments of wireless networks that are expected in the future, where numerous nodes may contend for the channel.

Obviously, increasing investment in preventive measures can render the corrective measures less expensive. The use of low complexity techniques for authentication, while reducing the cost of corrective measures, can also reduce the cost of preventive measures, as they help substantially towards realization of *inexpensive* trustworthy devices. Furthermore, increasing reliance on storage to improve security, apart from being inexpensive, does not in any way *hinder our ability* to improve trustworthiness.

For A-RPS, we saw that we can use storage in order to ensure that it is impractical for an attacker to "break the KPS." However, just the fact that "breaking the KPS" may be impractical is not a sufficient reason to settle for *ineffective* assurances. After all, in a scenario where an attacker has compromised secrets from 100 nodes (even if does not help the attacker break the security of A-RPS), the attacker can still wreak considerable damage with the secrets of 100 nodes. Using such secrets the attacker can send well authenticated "information," however misleading they may be.

Thus even if security solutions that are *not* susceptible to collusions are used, assurances for protection of secrets are still mandatory. It is important to see that assurances of read-proofing are needed not because we *need* to use KPSs. It is because such assurances are needed in *any* case. The use of PKPSs like A-RPS and A-HARPS can take advantage of such assurances to reduce overheads, and synergistically help in *improving our ability to realize such assurances* by mandating very low computational complexity.

VI. Conclusions

We presented a family of probabilistic "asymmetric" key distribution schemes employing different sets of secrets for authentication / encryption and decryption / verification, for mutual authentication and multi-source broadcast authentication.

One of the very desirable features of the proposed schemes is that they can advantageously utilize inexpensive resources like storage to reduce reliance on more more expensive resources like computation and bandwidth. Specifically, we demonstrated how the security of mutual authentication using A-RPS can be increased to a very large extent - large enough to render the problem of "lack of collusion resistance" irrelevant - without increasing the computational complexity. We also showed that the security of multi-source broadcast authentication schemes employing A-RPS or A-HARPS can be improved by increasing storage, without increasing the bandwidth required for the appended message authentication codes.

Limiting the computational complexity for evaluation of security associations is necessary for efficient realization of trustworthy devices. Further, for ad hoc networks, it is also important to keep the bandwidth overheads low. We argued that proposed schemes synergistically reduce the cost of preventive and corrective security measures for ad hoc networks.

References

[1] 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.

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

[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] S. M. Matyas, C. H. Meyer, "Generation, Distribution and Installation of Cryptographic Keys," IBM Systems Journal, **2**, pp 126 – 137, 1978.

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

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

[7] 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.

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

[9] L. Eschenauer, V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," Proceedings of the Ninth ACM Conference on Computer and Communications Security, Washington DC, pp 41-47, Nov 2002.

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

[11] M. Ramkumar, N. Memon, "An Efficient Random Key Predistribution Scheme for MANET Security," IEEE Journal on Selected Areas of Communication, March 2005.

[12] F. M. Williams, N. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Press, 1977.

[13] S.W. Smith, S. Weingart, "Building a High-Performance Programmable Secure Coprocessor," IBM Technical Report RC21102, Feb 1998.

[14] J.D Tygar, B. Yee, "Dyad: A system for Using Physically Secure Coprocessors," Technological Strategies for the Protection of Intellectual Property in the Networked Multimedia Environment, pp 121–152, 1994.

[15] R. Anderson, M. Bond, J. Clulow, S. Skorobogatov, "Cryptographic Processors - a survey," University of Cambridge, Computer Laboratory Technical Report, UCAM-CL-TR-641, Aug 2005.

[16] M.G Karpovsky, K. Kulikowski, A. Taubin, " Robust Protection Against Fault-Injection Attacks of Smart Cards Implementing the Advanced Encryption Standard". T Proc. Int. Conference on Dependable Systems and Networks (DNS 2004), July, 2004.