Broadcast Authentication With Hashed Random Preloaded Subsets

Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS 39762.

Abstract

We introduce a novel cryptographic paradigm of broadcast authentication with "preferred" verifiers (BAP). With BAP, the message source explicitly targets a set of one or more verifiers. For an attacker, forging authentication data of a source, for purposes of fooling preferred verifiers may be substantially more difficult than fooling other (non-preferred) verifiers. We investigate broadcast authentication (BA) with *hashed random preloaded subsets* (HARPS), which caters for such a distinction. HARPS, provides for efficient broadcast authentication, with and without preferred verifiers.

1 Introduction

A broadcast authentication scheme, permits any node to verify the authenticity of the source of the broadcast. This can be achieved using digital signatures if public key cryptography is used, or if only symmetric cryptography is used, by appending verifiable authentication data, consisting of multiple (shared-secrets based) message authentication codes (MAC) [2] - [4] to the message, such that *any* verifier would be able to verify at least a *subset* of the appended MACs. In the rest of this paper, we shall simply refer to the appended MACs (or authentication data) as $\mathfrak{A}$. More specifically, we shall represent the authentication data of a source i, for a message M as $\mathfrak{A}_i(M)$.

The *strength*, or the security offered by a broadcast authentication scheme is a measure of the difficulty an attacker faces, in forging $\mathfrak{A}$ of an arbitrary source, in order to fool arbitrary verifiers(s). The *complexity* of a scheme is a function of the *number* of appended MACs (the bandwidth needed for $\mathfrak{A}$). The *efficiency* of a broadcast authentication scheme is then a ratio of the strength to its complexity.

In this paper we consider broadcast authentication, using a random key pre-distribution scheme, hashed random preloaded subsets (HARPS) [1]. We show that broadcast authentication using HARPS is substantially more efficient than the schemes in [2] - [4].

While the purpose of broadcast is to reach every possible recipient, in real world applications each broadcast may have a different amount of "significance" to different recipients. For instance in applications involving multi-hop ad hoc networks [5], where for instance routing information in each node may be broadcast to the entire network, malicious broadcasts (with forged authentication) is more likely to affect the nodes in the immediate neighborhood than nodes further away. Under such circumstances, it would be useful if broadcast authentication could cater for the "higher strength" required for some "preferred" verifiers.

In addition to the providing more efficient general purpose broadcast authentication (henceforth referred to as BA) than the schemes in [2] - [4], HARPS supports this paradigm of "preferred" verifiers (henceforth referred to as BAP). When a broadcast is targeted to one or more preferred verifiers, the appended authentication data may be considerably more difficult for an attacker to forge, for the purpose of fooling (any of) the preferred verifiers.

BAP bears some resemblance to "signatures with *designated* verifiers" [6], [7]. However, while designated signatures can be verified *only* by designated verifiers, BAP can be verified even by non-preferred verifiers (nodes which are *not* explicitly targeted). Moreover, as we shall demonstrate later, while BAP is substantially stronger than BA (against attempts to fool preferred verifiers), BAP is only *marginally* weaker than general-purpose BA against attempts to fool other *non-preferred* verifiers.

The rest of this paper is organized as follows. In Section II we briefly review key pre-distribution (KPD) schemes, and more specifically, KPD schemes based on "preloaded subsets," with further emphasis on schemes employing "*random* preloaded subsets" (RPS). This is followed by a brief introduction[1] to HARPS (*hashed* RPS) [1]. In Section III we analyze the efficacy of BA using RPS (which is the basis for the schemes in [2] - [4]), and HARPS, and BAP using HARPS.

2 Key Pre-distribution

A KPD scheme consists of a trusted authority (TA), and N nodes with unique IDs. The TA chooses P secrets $\mathcal{R}$ and two operators $f()$ and $g()$. The operator $f()$, is used to determine the secrets $\mathbb{A}$ that are preloaded in node A. Two nodes with IDs A and B, with

[1] Ref. [1] has not been published yet. A brief introduction to HARPS makes this paper self-contained and preserves author anonymity. Further, the focus of Ref. [1] was the security of pair-wise and group secrets, and renewability of HARPS.

preloaded secrets $\mathbb{A}$ and $\mathbb{B}$ can discover a unique shared secret K_{AB} using a *public* operator $g()$ without further involvement of the TA. The restrictions on the operators $f()$ and $g()$ in order to satisfy these requirements can be mathematically stated as follows:

$$\mathbb{A} = f(\mathcal{R}, A), \text{ and } K_{AB} = g(\mathbb{A}, B) = g(\mathbb{B}, A) = f(\mathcal{R}, A, B) = f(\mathcal{R}, B, A). \quad (1)$$

As $g()$ is public, it possible for two nodes, just by exchanging their IDs, to execute $g()$ and discover a unique shared secret. As the shared secret is a function of their IDs, their ability to arrive at the shared secret provides mutual assurances to A and B that the other node possesses the necessary secrets $\mathbb{B}$ and $\mathbb{A}$, respectively. The secrets preloaded in each node is referred to as the node's *key-ring*. We shall represent by k, the size of the key ring.

The primary advantage of KPDs is their ability to cater for ad hoc authentication without active involvement of a trusted authority, and without employing asymmetric cryptography. However, this advantage comes at a price. Note that in KPD schemes, the keys assigned to different nodes are *not independent* - they are all derived from the same set (TA's) keys $\mathcal{R}$. Thus an attacker who has exposed keys from a *finite* number of nodes could compromise the entire system. For conventional key distribution schemes (KDS) (like Kerberos [8], [9] or PKI [10]) however, as the keys assigned to different nodes are *independent*, this is not the case.

However, for evolving [11] application scenarios (like MANETS [5]) where extensive *mutual co-operation* of resource constrained (battery operated) nodes is necessary for their very functioning, compromise of a few nodes could affect the entire deployment. Thus there is a need to take proactive steps to control sizes of *attacker coalitions* (perhaps by improved technology for tamper resistance / read-proofing of devices [12]). For securing such deployments, conventional KDSes may be an "overkill" (if the entire deployment is affected if a finite number of nodes are compromised, the fact that the *KDS* is not compromised does not help much). Thus KPDs, due to their inherent advantages of low resource consumption (which is also necessary as deployments of wireless devices forming MANETS are expected to include resource constrained devices), may be sufficient for securing such networks. This is perhaps the reason for renewed interest in KPD schemes in the recent past.

2.1 Preloaded Subset Key Distribution Schemes

KPD schemes based on the concept of pre-loading subsets[2] (say of cardinality k) of keys in each node, from a *pool* of P keys, has been employed by various researchers, for very different cryptographic primitives. Perhaps the earliest example is the matrix [13] key pre-distribution scheme by Gong et. al. The applications employing preloaded subsets range from discovery of shared secrets for pairwise communications [13] - [19], and more general group communications [1], [20], broadcast encryption [21], [22] and broadcast authentication [2] - [4].

While the earlier methods based on preloaded subsets favored deterministic allocation of keys to nodes (most of them perhaps motivated by Erdos et. al's seminal work on intersections of finite sets [23]), Dyer et al [3] was perhaps the first to point out the advantages of *random* allocation of subsets. A very elegant framework for analysis of the security of random preloaded subsets was also presented in [3]. Recent attempts in this direction too, [1], [14] - [20] favor random [14], [15], [19] or pseudo-random [1], [16], [20], allocation of keys (in this paper we shall collectively refer to them as RPS or random preloaded subsets). While all RPS based methods are essentially similar, the primary advantage of the methods which employ *pseudo-random* allocation of subsets, is that they provide a simple and elegant way for nodes to determine shared secrets (methods based on purely random allocation on the other hand need a bandwidth intensive shared key discovery process).

Formally, a (P, k) RPS employs a TA who chooses an indexed set of P keys $K_1 \cdots K_P$. Each node has a unique ID. The TA chooses public random function $F_{RPS}()$, which when "seeded" by a node ID, yields the allocation of keys for the node. Thus for a node A (node with unique ID A)

$$F_{RPS}(A) = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}, \text{ and } \mathbb{A} = \{K_{\alpha_1}, \ldots, K_{\alpha_k}\}. \quad (2)$$

where $1 \leq \alpha_i \leq P, \alpha_i \neq \alpha_j$ for $i \neq j$. In other words $F_{RPS}()$ generates a *partial* random permutation of $\{1 \cdots P\}$. The k-length sequence $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ is the index of the keys preloaded in node A (or node with ID A). $\mathbb{A}$ is the set of secrets preloaded in A. Note that the indexes are public (as the node ID and $F_{RPS}()$ are public).

KPD schemes employing random preloaded subsets fall under the category of *random* KPD schemes. However, the first random KPD scheme, LM [24], proposed by Leighton and Micali, employs a very different idea. In the (k, L) LM scheme, the TA chooses an indexed set of k secrets $K_1 \cdots K_k$, a cryptographic hash function $h()$, and a public random function $F_{LM}()$. For a node A,

$$F_{LM}(A) = \{a_1, a_2, \ldots, a_k\}, 1 \leq a_i \leq L \forall i. \text{ and } \mathbb{A} = \{^{a_1}K_1, ^{a_2}K_2, \ldots, ^{a_k}K_k\}. \quad (3)$$

In other words $F_{LM}()$ generates a k-sequence of uniformly distributed random integer values between 1 and L. The node A is preloaded with k keys. The i^{th} preloaded key is node A is derived by repeatedly hashing i^{th} TAs key a_i times. The parameter L is the maximum hash depth. The notation ^iK_j represents the result of *repeatedly* hashing of K_j, i times, using a (public) cryptographic hash function $h()$.

[2] For such schemes the function $f()$ in Eq (1) simply chooses a subset of keys, and the function $g()$ obtains the intersection of two subsets.

In HARPS [1], we proposed a random KPD scheme which is a generalization of LM and RPS. In (P, k, L) HARPS, the TA chooses P keys $K_1 \cdots K_P$, and each node is loaded with a *hashed* subset of k keys. The TA has an indexed set of P secrets, a cryptographic hash function $h()$ and a public random function $F_{HARPS}()$. For a node A,

$$F_{HARPS}(A) = \{(\alpha_1, a_1), (\alpha_2, a_2), \ldots, (\alpha_k, a_k)\}, \text{ and } \mathbb{A} = \{{}^{a_1}K_{\alpha_1}, {}^{a_2}K_{\alpha_2}, \ldots, {}^{a_k}K_{\alpha_k}\}. \tag{4}$$

The first coordinate $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ represents the index of the keys chosen to be preloaded in node A, and the second coordinate $\{a_1, a_2, \ldots, a_k\}$, the number of times each chosen key is hashed (using cryptographic hash function $h()$) before they are preloaded in the node A. Note that LM and RPS are actually special cases of HARPS. LM is HARPS with $P = k$, and RPS is HARPS with $L = 0$ (or keys are not hashed before pre-loading).

3 Broadcast Authentication with Preloaded Subsets

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 message authentication codes (MAC) - one corresponding to each of the k keys it possesses (or P keys if the source is the TA). Any node will be able to verify the authenticity of the broadcast by checking the MACs corresponding to *all* the keys the verifying node shares with the message source.

It is assumed that the KPD secrets (HARPS / RPS *key-rings*) are stored in "tamper-resistant" and "read-proof" devices. However, an attacker with sufficient resources may be able to compromise the secrets stored in *some* devices.

For BA using RPS, employing a cryptographic hash function $h()$, the authentication data of a node A (with keys $\mathbb{A} = \{K_{\alpha_1} \cdots K_{\alpha_k}\}$), for a message M is

$$\mathfrak{A}_A(M) = [H_1 \parallel H_2 \parallel \cdots \parallel H_k], \quad H_i = h(A \parallel M \parallel K_{\alpha_j}), 1 \leq j \leq k \tag{5}$$

The primary distinction between BA-using-RPS (as in [2] - [4]) and BA-using-HARPS, is that in the latter, the source has the added flexibility of *choosing the hash depth* of the keys used for MACs - the source can choose any hash depth equal to or greater than the depth it possesses for that particular key. For example, if a node has the i^{th}key at a hash depth a (or the key ${}^{a}K_i$), the node can use any ${}^{x}K_i$ as the corresponding MAC key, where $a \leq x \leq L$.

For BA using HARPS, the authentication data of a node A (with keys $\mathbb{A} = \{{}^{a_1}K_{\alpha_1}, {}^{a_2}K_{\alpha_2}, \ldots, {}^{a_k}K_{\alpha_k}\}$), for a message M is

$$\mathfrak{A}_A(M) = [H_1 \parallel H_2 \parallel \cdots \parallel H_k], \quad H_i = h(A \parallel M \parallel {}^{x_j}K_{\alpha_j}), a_j \leq x_j \leq L, 1 \leq i \leq k. \tag{6}$$

If the source chooses the maximum hash depth L (or $x = L$) for every key, then *any* verifier who has the i^{th}key will be able to verify the MAC. If a source node chooses to use some other value of hash depth, say $x < L$, it may result in some nodes not being able to use that key for verification, even though the verifying node has a key corresponding to that index (the verifying node may have key ${}^{y}K_i$ with $y > x$). However, it is also less likely that a coalition of attackers could forge the MAC.

What we desire is to reduce the *probability* p_F, that an attacker who has exposed keys belonging to n nodes, can forge a message for the purpose of fooling a verifier (lower the probability p_F, higher the strength of the authentication scheme). Obviously, for a given P, k, L and n, there is an *optimal* choice of x - which we shall represent by L_p. So a simple strategy is to choose the optimal hash depth L_p - whenever possible. For keys in source node where the hash depth $d > L_p$ the source node *cannot* use depth L_p. It has to use the minimum possible hash depth it can - which is the current depth d of the key.

However, once a *strategy* is fixed, 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) knows what hash depths have been (or should be) used by the source for arriving at $\mathbb{A}$.

The primary reason for the choice of including[3] the ID of a node for calculation of the $H_i, 1 \leq i \leq k$ in the authentication data $\mathfrak{A}(M)$ is to ensure that the attacker cannot "pool" authentication data from different nodes (say $B_1 \cdots B_n$) for the same message M to forge $\mathfrak{A}_A(M)$. The only way for the attacker to forge messages is by actually tampering with devices, and exposing buried secrets.

There are four different scenarios of broadcast authentication to be considered

1. broadcast by TA, verification by nodes.
2. broadcast by a node, verification by TA.
3. broadcast by a node, verification by a peer node.
4. broadcast by a node, joint verification by J peer nodes.

In the first scenario, the source is the TA, and $\mathfrak{A}_{TA}(M)$ (for some message M) has P MACs. In all other scenarios, the source is a node with k preloaded secrets, and hence $\mathfrak{A}$ has k MACs. For scenario 2, the verifier is the TA - or equivalently, a verifier would not

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

accept $\mathfrak{A}$ as authentic unless the TA assures it as authentic. As the TA would be able to verify *every* MAC appended by *any* node, this would be the most challenging of scenarios for any adversary. Unfortunately, in many application scenarios, it might not be possible for nodes to have access to the TA in order to verify $\mathfrak{A}$. In scenario 3, a node takes on complete responsibility for verifying $\mathfrak{A}$. However, in many cases it may possible for the verifier to check the authenticity of the broadcast with few *other* nodes. In other words, the verifier would accept $\mathfrak{A}$ as authentic only of it can verify its authenticity *and* $J-1$ other nodes confirm the authenticity by verifying the MACs (or J nodes *jointly* verify $\mathfrak{A}$).

For scenarios 3 and 4 we also consider the possibility of broadcast authentication with preferred verifier(s) (or BAP). For BAP (from source A, for a message M) we represent the authentication data by $\mathfrak{Ap}_A(M)$ (the ID(s) of preferred verifier(s) is (are) not explicitly included in the notation). In the rest of this section we shall look at each of these scenarios is detail.

3.1 Summary of Notations

As each node is preloaded with k out of P keys, and probability that the key with index i is chosen for any node (or preloaded in any node) is $\xi = \frac{k}{P}$. Let $B_\xi(n,u)$ represent the binomial probability that *exactly* u out of n nodes have the key with index i. Let $G_L(d)$ represent the probability that the hash depth of any key is greater than d. Thus we have

$$\xi_i = \frac{k}{P} = \xi \forall i, \quad B_\xi(n,u) = \binom{n}{u}\xi^u(1-\xi)^{n-u}, \quad G_L(d) = \frac{L-d}{L}. \tag{7}$$

To distinguish between the four scenarios we shall use the following suffixes:

$_1$: TA $\rightarrow$ node $\quad$ $_2$: node $\rightarrow$ TA $\quad$ $_3$: node $\rightarrow$ node $\quad$ $_4$: node $\rightarrow$ jointly verifying nodes

To distinguish HARPS and RPS (for **BA**) we use suffixes R and H. For **BAP** with HARPS we use the suffix P. However, when BAP is verified by non-preferred verifiers the suffix P' is used instead.

$_R$: BA (RPS) $\quad$ $_H$: BA (HARPS) $\quad$ $_P$: BAP for preferred verifiers $\quad$ $_{P'}$: BAP for *other* verifiers

For example, $_{R3}$ represents BA using RPS, where the source and verifier are peers. As another example $_{P'4}$ represents the case when the source is a peer, and the BAP meant for "joint verification by some preferred nodes", is actually verified by a non-preferred node.

We shall also represent by ϵ, the probability that a particular MAC (say corresponding to key index $i, 1 \le i \le P$) is "safe" - which happens when

- the source has the i^{th}key. *and*
- the verifier can verify the MAC, *and*
- the attacker (coalition of n nodes) *cannot* forge the MAC.

In order to be successful, the attacker needs to forge *every* MAC that a verifier can verify. We shall represent by p_F, the probability of successful forgery by an attacker.

3.2 Broadcast by TA - Verification by Nodes

For RPS, the TA appends P MACs to the message. Any node can verify k of the P MACs. The MAC corresponding to any key (say index i) is "safe" if the verifier has the key (which happens with probability ξ) and if the coalition of n nodes (nodes whose keys have been exposed by an attacker) *do not* have the i^{th} key (probability $(1-\xi)^n$). The probability ϵ_{R1} that the i^{th} is safe, and the probability p_{FR1} that the attacker coalition can successfully impersonate the TA for the purpose of fooling any node is now

$$\epsilon_{R1}(n) = B_\xi(n,0) = \xi(1-\xi)^n, \quad p_{FR1}(n) = (1-\epsilon_{R1}(n))^P. \tag{8}$$

For HARPS, the TA has the ability to choose *any* hash depth. If the TA chooses a strategy of choosing hash depth of L_p for each MAC key, any node can verify the i^{th} key with a probability $\xi\frac{L_p}{L}$ (the probability that a verifier has the key is ξ, and the probability that the hash depth of the key is not greater than L_p is $\frac{L_p}{L}$. The probability that exactly u of n nodes have the i^{th} key is $B(n,u,\xi)$, and the probability that all u keys have hash depth greater than L_p is $G_L(L_p)^u$. Thus,

$$\epsilon_{H1}(n) = \left(\xi\frac{L_p}{L}\right)\sum_{u=0}^{n} B_\xi(n,u)G_L(L_p)^u, \tag{9}$$

and $p_{FH1}(n) = (1-\epsilon_{H1}(n))^P$. Figure 1 (Lines 1 to 4) depicts the relationship between p_F and n (probability of successful forgery by a coalition of attackers who have exposed all keys from n nodes), for $L_p = 64, 32, 16, 8$ respectively. For the y-axis we use $-\log_2(p_F)$ as the scale - which could be considered as "bit-security." For example $p_F \approx 10^{-20}$ is roughly equivalent to 64-bit security - $\frac{1}{2^{64}} \approx 10^{-20}$ is the probability with which one can successfully "guess" a 64 bit key! For the plots we have chosen $P = 18,000$, $k = 1500$ and $L = 64$

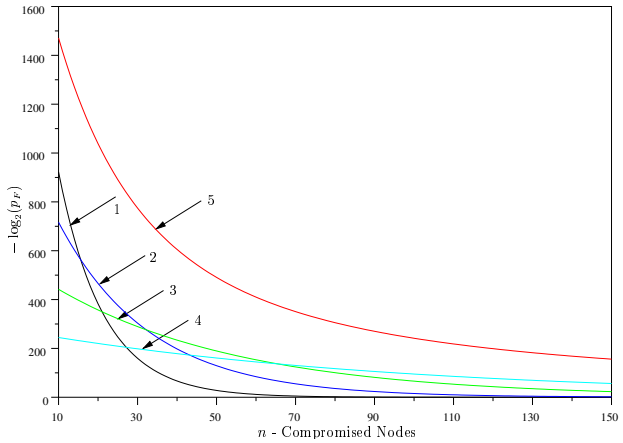

Figure 1: Lines 1 to 4: Broadcast by TA and verification by nodes for $L_p = L = 64, L_p = 32, L_p = 16, L_p = 8$ respectively. The case of $L_p = L = 64$ (Line 1) also corresponds to RPS. Further Line 1 also corresponds to the case of broadcast by node and verification by TA, using RPS. Line 5: Broadcast by a node, verification by TA for HARPS. For all plots $P = 18000, k = 1500$ and $L = 64$.

For purposes of broadcast authentication, the performance of RPS ($L = 0$) is identical to the case of HARPS with $L_p = L = 64$. This can be easily verified by substituting L_p by L in Eq (9). The plots indicate that a choice of large L_p (closer to L) offers higher bit-security for *small* values of n. However, for larger values of n choosing smaller and smaller values of L_p does better. In practice lower values of L_p may be more useful. The fact that $L_p = 8$ offers a bit-security of about 250 bits for $n = 10$ is good enough. That RPS or ($L_p = L$) offers over 900 bits is probably an "overkill". However, RPS is insecure for $n > 70$ while HARPS with $L_p = 8$ is still very much usable when $n \approx 150$.

Obviously, for a given n, and a restriction on available resources (primarily k - we do not care too much about P, the number of keys that the TA has to store - which is also the bandwidth of authentication data produced by the source) one could simultaneously optimize the values of L_p and $\xi = k/P$. We could consider n as a measure of the expected "threat level" for the deployment. However, once deployed, it is no longer possible to *change* ξ (or P and k). And the deployment is expected to operate safely at different threat levels. The choice of the parameters $P = 18,000$ and $k = 1500$ for illustrative purposes is arbitrary[4].

One of the biggest advantages of HARPS over RPS is that the hash depth used for MACs can be changed *post-deployment*! In other words, depending on the threat level (perceived value of n) the TA could choose *optimal* values of L_p (which is obviously not possible with RPS). As the threat level increases, the source could choose lower and lower hash depths.

3.3 Broadcast by a Node - Verification by TA

In this case, the TA can verify each of the k MACs appended by the node. For RPS, it can be very easily seen that the situation is exactly similar to case 1. Thus $p_{FR2} = p_{FR1}$. However, for HARPS, the situation is a little different. Unlike broadcasts by TA where the TAs best strategy is to choose a fixed hash depth for all keys, for broadcasts by nodes (and verification by TA) the nodes would choose the *lowest possible hash depth* for each key - which would be the same as the hash depth of their preloaded keys. The probability that the source has the i^{th}key is ξ. The probability that the depth of the i^{th}key is l is $\frac{1}{L}$. So we have

$$\epsilon_{H2}(n) = \xi \sum_{u=0}^{n} B_\xi(n,u) \frac{1}{L} \sum_{l=1}^{L} G_L(l)^u, \tag{10}$$

and $p_{FH2}(n) = (1 - \epsilon_{H2}(n))^P$. Line 5 in Figure 1 is a plot of $-\log_2(p_F)$ vs n for HARPS (line 1 for RPS as $p_{FR2} = p_{FR1}$). A comparison of lines 1 and 5 in Figure 1 shows that HARPS is *substantially* better than RPS in this case.

3.4 Broadcast by a Node - Verification by a Peer

In this case, the source, has the i^{th}key with probability ξ. However, the verifier being another node also has the key with probability ξ. So, in order for the i^{th}key to be safe, both the source and verifier should have the i^{th}key and the n attackers together should *not* have the i^{th}key. Thus, for RPS, $\epsilon_{R3}(n) = \xi^2(1-\xi)^n$, and $p_{FR3}(n) = (1 - \epsilon_{R3}(n))^P$.

For HARPS, the source's strategy, as in case 1, is to choose an optimal hash depth of L_p, whenever possible.

[4]But this choice is still optimal for *some* n!

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 $\frac{l}{L}$. Thus

$$\epsilon_{H3}(n) = \left(\xi \frac{L_p}{L}\right)^2 \sum_{u=0}^{n} B_\xi(n,u) G_L(L_p)^u + \sum_{l=L_p+1}^{L} \xi^2 \frac{l}{L^2} \sum_{u=0}^{n} B_\xi(n,u) G_L(l)^u, \tag{11}$$

and $p_{FH3}(n) = (1 - \epsilon_{H3}(n))^P$.

3.4.1 BAP with T Preferred Verifiers:

However, for HARPS, there is an additional possibility - targeting the authentication data to *preferred* verifiers (BAP). In this case the authentication data would also consist of the list of preferred verifiers. For example, for a case where T nodes $V_1 \cdots V_T$ are designated as preferred verifiers, we have

$$\mathfrak{Ap}_A(M) = [V_1 \parallel \cdots \parallel V_T \parallel H_1 \parallel H_2 \parallel \cdots \parallel H_k], \quad H_i = h(V_1 \parallel \cdots \parallel V_T \parallel A \parallel M \parallel {}^{x_j}K_{\alpha_j}). \tag{12}$$

Under this condition, the broadcasting node may use a strategy[5], where the hash depth is chosen such that the preferred node(s) can verify as many MACs as possible. For a scenario where a source explicitly targets T preferred nodes, the strategy for choosing the hash depths would be to choose the maximum of the hash depths of the source and the T verifiers for each shared key. For example, for the i^{th}key, if the hash depth of source is s and if j out of T verifiers have the i^{th}key with hash depths $v_1 \cdots v_j$, then the source would choose the depth as $\max(s, v_1 \cdots v_j))$.

The probability that the source, and exactly j of the T verifiers have the i^{th}key is $\xi B_\xi(T, j)$. Under this condition, the source would choose a depth l for the i^{th}key if

1. Source has hash depth l (probability $\frac{1}{L}$) and all j (out of T) verifiers have hash depths less than or equal to l (probability $(1 - g(l))^j = \frac{l^j}{L^j}$), or
2. Source has hash depth less than l (probability $\frac{l-1}{L}$) and j verifiers have a maximum hash depth equal to l (probability $\frac{l^j - (l-1)^j}{L^j}$)

We shall represent the conditional probability of choosing depth l as $P_j(l)$, where

$$P_j(l) = \frac{1}{L} \frac{l^j}{L^j} + \frac{(l-1)}{L} \frac{\left(l^j - (l-1)^j\right)}{L^j}. \tag{13}$$

Also note that under the condition that j of T verifiers have the i^{th}key, the probability that a *particular verifier* has the I^{th}key is $\frac{j}{T}$. Thus

$$\epsilon_{P3}(n, T) = \xi \sum_{j=1}^{T} \frac{j}{T} B_\xi(T, j) \sum_{l=1}^{L} P_j(l) \sum_{u=0}^{n} B_\xi(n,u) G_L(l)^u. \tag{14}$$

The plots for RPS and HARPS for peer-to-peer broadcast authentication, with different values of L_p are shown in Figure 2. For general purpose BA (Lines 1 to 4 in the figure), similar to case 1 (broadcast by TA and verification by nodes), lower values of L_p are more useful for larger n .

Line 5 is the plot for BAP with HARPS with one preferred recipient ($T = 1$). Note that it is substantially more difficult for attackers to fool the preferred verifier (compare line 5 with lines 1,2,3,4). As T increases, $\mathfrak{Ap}$ becomes less and less "targeted." Line 6 is the plot for $T = 2$ (which is marginally weaker than the case of $T = 1$, but however indistinguishable in the Figure). The difference is more readily apparent for larger values of T (as the targeting gets less "specific" the strength gain due to targeting reduces). Line 7 is the plot for $T = 20$. As $T \to \infty$ (which defeats the purpose of targeting anyway) the source would be forced to choose the maximum hash depth for each key - or the situation is similar to that of choosing $L_p = L$ (line 1, RPS).

Thus HARPS provides for two mechanisms for improving the strength of broadcast authentication - the ability to choose L_p dynamically depending on the expected threat level (which improves the security of BA), and the ability to explicitly target verifiers (BAP), which yields substantial gains over BA. Note BAP is substantially more secure over *all* n.

3.5 Broadcast by a Node - Joint Verification by J Peers

In this case, the attacker coalition has to fool *all* J nodes in order to be successful. For RPS, the i^{th}key is safe when the source and *any* of the J verifiers has the i^{th}key. The probability that at least one of the J verifiers have the i^{th}key is $(1 - (1 - \xi)^J$. Thus $\epsilon_{R4}(n) = \xi(1 - (1 - \xi)^J)(1 - \xi)^n$, and $p_{FR4}(n) = (1 - \epsilon_{R4}(n))^P$.

[5]The context of the application would dictate whether the source should have chosen preferred verifiers for a particular message M, through regulated "policies."

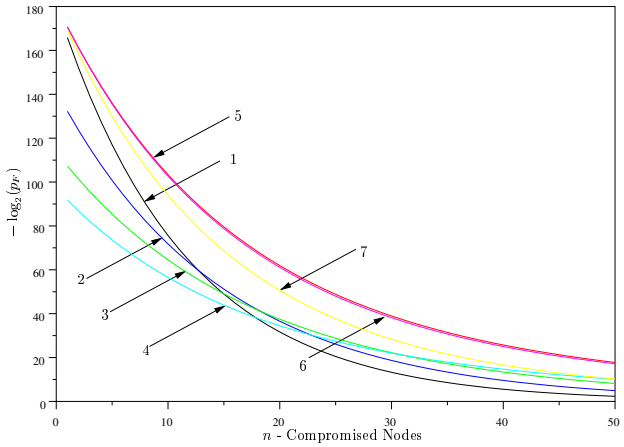

Figure 2: Broadcast by nodes and verification by a peer. Line 1: $L_p = L = 64$ (also RPS). Lines 2 to 4: $L_p = 48$, $L_p = 32$ and $L_p = 16$ respectively. Line 5: $\mathfrak{Alp}$ targeted at one verifier ($T = 1$). Line 6: $\mathfrak{Alp}$ targeted to $T = 2$ verifiers - (lines 5 and 6 are indistinguishable in the figure). Line 7: $T = 20$.

In the case of HARPS, once again there are two different approaches the source could take - depending on whether the source knows apriori, the identities of the J (jointly) verifying nodes. For BA, the same strategy of fixed L_p can be used.

The probability that exactly j out of J verifiers have the i^{th}key is $B(J, j, \xi)$, and the probability that at least one of the j keys has hash depth less than or equal to l is $1 - G_L(l)^j$. Thus

$$\begin{aligned}\epsilon_{H4}(n) &= \xi \frac{L_p}{L} \sum_{j=1}^{J} B_\xi(J, j)(1 - G_L(L_p)^j) \sum_{u=0}^{n} B_\xi(n, u) G_L(L_p)^u \\ &+ \frac{\xi}{L} \sum_{l=L_p+1}^{L} \sum_{j=1}^{J} B_\xi(J, j)(1 - G_L(l)^j) \sum_{u=0}^{n} B_\xi(n, u) G_L(l)^u, \end{aligned} \tag{15}$$

and $p_{FH4}(n) = (1 - \epsilon_{H4}(n))^P$.

3.5.1 BAP with J Preferred Joint Verifiers:

For BAP, in this case the identities of the preferred recipients (who *jointly* verify $\mathfrak{Alp}$), are known apriori. Once again the source node chooses the hash depths to "suit" the J verifiers. In this case, the strategy would be to choose the minimum possible hash depth such that at least *one* of the J verifiers can verify the MAC[6]. For example, for the i^{th}key, if the hash depth of the source is s and if j out of J verifiers have the i^{th}key with hash depths $v_1 \cdots v_j$, then the source would choose the hash depth as $\max(s, \min(v_1 \cdots v_j))$

The source can use the i^{th}key with probability ξ, and the probability that j out of J verifiers have the i^{th}key is $B(J, j, \xi)$. Under this condition, the source would choose a depth l for key i under two conditions:

1. Source has depth l (probability $\frac{1}{L}$) and the minimum depth of j nodes for key i (which have key i) is less than or equal to j - which happens with probability $(1 - G_L(l)^j)$.
2. Source has depth less than l (probability $\frac{l-1}{L}$)) and the minimum hash depth for the i^{th}key among the j nodes is *exactly* l. The probability that the minimum depth is greater than $l - 1$ is $G_L(l - 1)^j$, and the probability that the minimum depth is greater than l is $G_L(l)^j$. Thus the probability that the minimum depth is equal to l is $G_L(l - 1)^j - G_L(l)^j$.

Let $Q_j(l)$ denote the conditional probability that the source chooses depth l for any key. We have

$$Q_j(l) = \frac{1}{L}\left(1 - G_L(l)^j\right) + \frac{l-1}{L}\left(G_L(l-1)^j - G_L(l)^j\right), \tag{16}$$

and

$$\epsilon_{P4}(n, J) = \xi \sum_{j=1}^{J} B_\xi(J, j) \sum_{l=1}^{L} Q_j(l) \sum_{u=0}^{n} B_\xi(n, u) G_L(l)^u \tag{17}$$

[6]This is the primary distinction between choice of hash depths for *individually* targeting T verifiers and *jointly* targeting J verifiers. For the former case, the hash depths are chosen to "reach" *every* preferred verifier that has the corresponding key. For the latter case the hash depth is chosen to reach *any* of them.

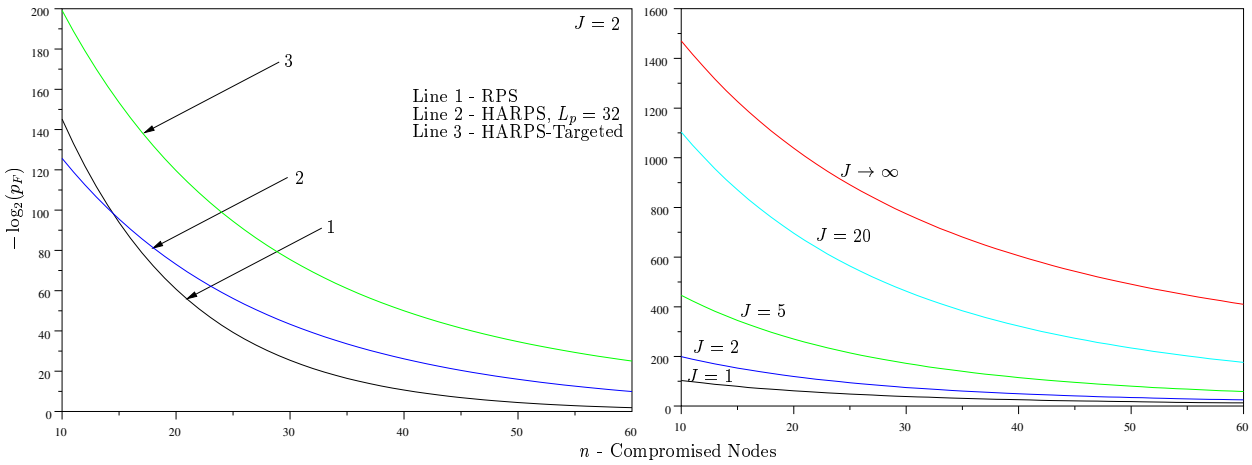

Figure 3: Broadcast by node and verification J peers. Left: $J = 2$ - line 1: RPS, line 2: BA with HARPS, line 3: BAP with HARPS. Right: BAP with HARPS for $J = 1, 2, 5, 20$ and $J \to \infty$.

The plots in Figure 3 (left) provide a comparison of RPS and HARPS (two cases - one with a fixed depth of $L_p = 32$ for BA, and the second for BAP), for $J = 2$. As expected, HARPS with $L_p < L$ performs better than RPS (or equivalently HARPS with $L_p = L$) especially for larger n. Also, HARPS "tuned" for known verifiers (line 3) performs significantly better. Figure 3 (right) plots $-\log_2(p_F)$ vs n for different values of J for BAP. Obviously, increasing the number of joint verifiers would make it more and more difficult for the attacker to forge the authentication data. In fact, if $J \to \infty$, the situation is not any different from case 2 - verification by TA.

3.6 Verification of BAP by *Other* Verifiers

Note that BAP only affects the hash depths of the keys that the source shares with the T preferred nodes (or J preferred joint-verifiers). For the other keys the source would employ the same tactic used for general purpose BA - a fixed "optimal" hash depth L_p (whenever possible). We already know that BAP is a lot "stronger" than BA, for the purpose of verification by *preferred* verifiers. The question now is, how does this affect verification of the BAP by other, non-preferred verifiers? Once again, we need to consider two different scenarios - $\mathfrak{Ap}$ targeted at T independent verifiers, and $\mathfrak{Ap}$ targeted at J joint verifiers.

For the case of verification by *other* verifiers of a BAP *individually targeted* to T verifiers, the expression for the probability that the i^{th}key is safe, is

$$\begin{aligned}\epsilon_{P'3}(n,T,L_p) &= \xi\sum_{j=1}^{T} B_\xi(T,j)\sum_{l=1}^{L} P_j(l)\frac{l\xi}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(l)^u \\ &+ \frac{\xi L_p}{L}B_\xi(T,0)\frac{\xi L_p}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(L_p)^u \\ &+ \frac{\xi}{L}B_\xi(T,0)\sum_{l=L_p+1}^{L}\frac{\xi l}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(l)^u. \end{aligned} \tag{18}$$

For the case of verification by other verifiers of a BAP *jointly targeted* to J verifiers, the expression for the probability that the i^{th}key is safe, is

$$\begin{aligned}\epsilon_{P'4}(n,J,L_p) &= \xi\sum_{j=1}^{J} B_\xi(J,j)\sum_{l=1}^{L} Q_j(l)\frac{l\xi}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(l)^u \\ &+ \frac{\xi L_p}{L}B_\xi(J,0)\frac{\xi L_p}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(L_p)^u \\ &+ \frac{\xi}{L}B_\xi(J,0)\sum_{l=L_p+1}^{L}\frac{\xi l}{L}\sum_{u=0}^{n} B_\xi(n,u)G_L(l)^u \end{aligned} \tag{19}$$

Note that the first terms (in both equations) account for the keys that are affected due to targeting. The second and third terms account for other keys - keys for which the source has a hash depth less than or equal to L_p (in which caser the source could employ

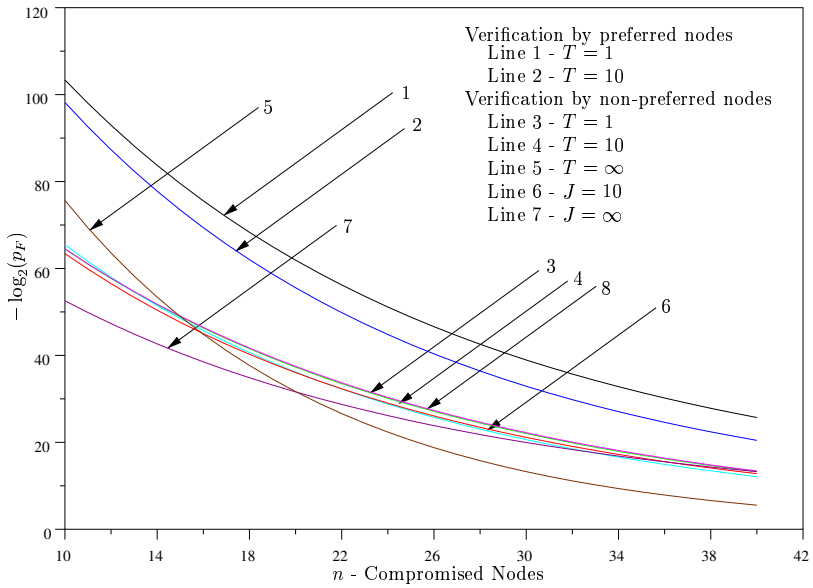

Figure 4: Verification by "non-preferred" verifiers. Line 1 - Verification by preferred verifier when $T = 1$. Line 2 - Verification by preferred verifier when $T = 10$. Lines 3 to 7 - Verification of $\mathfrak{Ap}$ by a non-preferred verifier. Line 8 - Verification of non-targeted (BA) $\mathfrak{A}$.

depth L_p) and keys for which the source has depth greater than L_p (in which case the source would use its hash depth for that key). As usual the probability $p_{FP'3}(n, T, L_p) = (1 - \epsilon_{P'3}(n, T, L_p))^P$, and $p_{FP'4}(n, J, L_p) = (1 - \epsilon_{P'4}(n, J, L_p))^P$.

Figure 4 depicts plots of the strength of the authentication for various cases.

Line 1 depicts the strength of BAP for a *preferred* verifiers ($T = 1$). Line 3, which is complementary to line 1 depicts the strength of BAP for other verifiers. It is no surprise that line 1 is significantly better than line 3. Similar plots for $T = 10$ are depicted by line 2 (for preferred verifiers) and its complement, line 4 (for other verifiers).

Line 8 is the strength of a (non-targeted) BA. Note that lines 3,4 are practically *indistinguishable* from line 8. This implies that there is practically no disadvantage to targeting - or BAP for other non-preferred verifiers is almost as good as general purpose BA!

Line 5 corresponds to a degenerate case of $T \rightarrow \infty$ - or if a *large* number of nodes are targeted (which once again, defeats the whole purpose of choosing preferred verifiers). Effectively, this is equivalent to RPS, or HARPS with $L_p = L$.

Line 6 is the for the case of BAP meant for *joint* verification by $J = 10$ preferred verifiers, verified however by a non-preferred verifier. Line 7 corresponds to the case of BAP targeted at $J =\rightarrow \infty$ jointly verifying preferred verifiers, verified by a non-preferred verifier. In this case the source would end up choosing the minimum possible hash depth for each key - or this is equivalent to BA with $L_P = 0$ (for the unintended verifiers).

4 Conclusions

We presented novel cryptographic paradigm of broadcast authentication with preferred verifiers (BAP), using hashed random preloaded subsets (HARPS). It was shown that BAP can be substantially stronger against attempts by attackers to fool preferred verifiers, than general-purpose BA using HARPS. Simultaneously BAP is only *marginally* weaker than BA against attempts to fool other verifiers. It was also quantitatively demonstrated that even for general purpose BA, the flexibility offered by HARPS to choose an optimal hash depth for the keys used for MACs, depending on the prevailing threat level, makes HARPS substantially stronger than RPS (or the schemes in [2] - [4]).

Note that while HARPS offers "freedom" to the source node for choosing hash depths, such freedom should be *regulated* by strict policies. In the absence of such policies an attacker would be able to choose the hash depths to make forgery simpler! The policies, could be regulated and disseminated (say using broadcast authentication!) periodically by the TA.

While we considered only peer-to-peer and peer-to"jointly-verifying-peers" scenarios for BAP, it is also possible for the TA to employ BAP, explicitly specifying T independent preferred verifiers or J jointly-verifying preferred verifiers. Further, for our analysis of jointly verified BAP, we assumed that all J joint verifiers are also preferred verifiers. In practice, it is also possible that only some of the joint verifiers are preferred verifiers. However, extensions of the analysis to all these cases are straight-forward.

Similar to the concept of explicitly targeting nodes, the source may also explicitly *exclude* nodes. This can be done by choosing hash depths to ensure that excluded nodes may not be able to verify the authentication data with high certainty (an excluded node will

still be able to verify MACs corresponding to keys it shares for the source node, for which it has a lower hash depth than the source node). This would increase the security of authentication against attackers when there are "reasons to believe" that some nodes may have been compromised (or in practice, this may be used to exclude nodes that have been "revoked" from the system because they are suspected to be compromised).

Bibliography

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

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

[5] Web Link, http://www.ietf.org/html.charters/manet-charter.html

[6] M. Jakobsson, K. Sako, R. Impagliazzo, "Designated Verifier Proofs and Their Applications," Lecture Notes in Computer Science, **1070**, pp 1070–1079, 1996.

[7] D. Chaum, "Designated Confirmer Signatures," Eurocypt 1994, LNCS 950, pp. 86–91. Springer-Verlag, 1995.

[8] B. C. Neuman, T. Ts'o, "Kerberos: An Authentication Service for Computer Networks", IEEE Communications, **32**(9), pp 33–38. September 1994.

[9] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," Communications of the ACM, 21(12), December 1978.

[10] S.Kiran, P. Lareau, S.Lloyd, "PKI Basics - A Technical Perspective," PKI-Forum (http://www.pkiforum.org), November 2002.

[11] J. Crowcroft, "Scalable and Ubiquitous Computing Systems," Grand Challenges in Computing (Research), edited by T. Hoare and R. Milner, 2004.

[12] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, "Tamper Proof Security: Theoretical Foundations for Security Against Hardware Tampering," Theory of Cryptography Conference, Cambridge, MA, February 2004.

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

[14] 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.

[15] H. Chan, A. Perrig, D. Song, "Random Key Pre-distribution Schemes for Sensor Networks," IEEE Symposium on Security and Privacy, Berkeley, California, May 2003.

[16] R. Di Pietro, L. V. Mancini, A. Mei, "Random Key Assignment for Secure Wireless Sensor Networks," 2003 ACM Workshop on Security of Ad Hoc and Sensor Networks, October 2003.

[17] S. Zhu, S. Xu, S. Setia S. Jajodia, "Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach," Proc. of the 11th IEEE International Conference on Network Protocols (ICNP'03), Atlanta, Georgia, November 4-7, 2003.

[18] W. Du, J. Deng, Y.S. Han. P.K.Varshney, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks," Proceedings of the 10th ACM Conference on Computer and Communication Security, pp 42–51, 2003.

[19] D. Liu, P.Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," Proceedings of the 10th ACM Conference on Computer and Communication Security, Washington DC, 2003.

[20] M. Ramkumar, N. Memon, R. Simha, "Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks," Globecom-2003.

[21] J. Garay, J. Staddon and A. Wool, “Long-Lived Broadcast Encryption,” Proceedings of Advances in Cryptology - CRYPTO 2000, Mihir Bellare (Ed.), LNCS (1880), Springer-Verlag, pp. 333-352, August 2000.

[22] C. Padro, I. Gracia, S. Martin, P. Morillo, “Linear Broadcast Encryption Schemes,” *Discrete Applied Mathematics*, **128**(1) pp 223–238, 2003.

[23] P. Erdos, P. Frankl, Z. Furedi, “Families of Finite Sets in which no Set is Covered by the Union of r Others,” *Isreal Journal of Mathematics*, **51**, pp 79–89, 1985.

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