Proxy Aided Key Pre-distribution Schemes for Sensor Networks

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

***Abstract*— Classical sensor network applications involve deployments of a large number of resource constrained wireless sensors in conjunction with a few "more capable" proxy devices. Many key distribution schemes employing only symmetric cryptographic primitives have been proposed for establishing pairwise secrets between sensors. At one end of the spectrum are Kerberos-like approaches where proxy devices are trusted with secrets of sensors, and mediate establishment of pairwise secrets between sensors. At the other end of the spectrum are many collusion susceptible key pre-distribution schemes that do not rely on the proxy devices for establishment of pairwise secrets. In this paper we propose a third approach where while the proxies aid the sensors in the process of establishment of pairwise secrets, the proxy devices are not trusted with the secrets of the sensors.**

I. Introduction

Ad hoc networks of spatially distributed battery powered wireless sensors will be useful for many application scenarios involving monitoring of environmental conditions like temperature, pressure, humidity, pollution levels, etc. Such networks are typically constituted by many inexpensive wireless sensors with limited battery resources and limited transmission range (to reduce their battery consumption) [1]. Sensors will relay measurements over multiple hops to one (or more) more capable proxy devices. The proxies may have intermittent access to satellite channels for relaying measurements to a remote location. $\propto$ Typically a large number sensors and a small number of proxies may be aerially dropped over a region to be monitored. The sensors and proxy devices may then engage in protocols to facilitate i) geographic localization of sensors [2] and ii) determination of optimal paths for relaying measurements to the closest proxy. In some application scenarios proxies may also be directed (from a remote location) to send specific queries to specific sensors. Sensors may also exchange measurements amongst themselves, possibly for more efficient relaying of sensed measurements.

A. Key Distribution

One important requirement for securing interactions between sensors is the ability to establish pairwise secrets between sensors. There are many compelling reasons to limit key distribution schemes for this purpose to only symmetric cryptographic primitives.

At one end of the spectrum of such schemes are Kerberos-like scheme where proxies mediate establishment of pairwise secrets between sensors [5]. At the other end of the spectrum are key predistribution schemes (KPS) that facilitate ad hoc establishment (without the need for a mediator) of pairwise secrets between sensors [6]-[9]. KPSs can be broadly classified into nonscalable and scalable schemes. For a network size of N nonscalable schemes require every node to store $\mathbb{O}(N)$ values. Scalable schemes which can support *unlimited* network size are however susceptible to collusions. Most n-secure scalable KPSs (which can resist collusions of up to n nodes) require every node to store $\mathbb{O}(n)$ values.

Due to the modest storage abilities of sensors,

1) nonscalable KPSs can only support limited network size; and
2) scalable KPSs (which can support unlimited network sizes) can only offer low levels of collusion resistance.

B. Contributions

In this paper we propose a third approach where while the proxies aid the process of establishing pairwise secrets, the proxies are not trusted with the secrets of the sensors. We propose two key predistribution schemes where the sensors can make effective use of the richer but untrusted resources of proxy devices to establish pairwise secrets. Consequently,

1) the first nonscalable scheme can realize "reasonably large" network sizes without the problem of susceptibility to collusions
2) the second scheme can support *unlimited* network size while achieving large collusion resistance.

In Section II we provide a brief overview of key predistribution schemes. The network model and assumptions made are enumerated. Current approaches for key distribution in classical sensor networks and their limitations are also described. In Section III we argue that proxies can aid the process of establishing shared secrets between sensors and describe some of the

desirable properties of key distribution schemes for this purpose. The two proposed schemes are also described in Section III. Conclusions are offered in Section IV.

The following notations are used in this paper

- $h()$: cryptographic hash function like SHA-1.
- $\mathbb{Z}_u$: the finite set of non-negative integers $\{0, 1, \ldots u-1\}$.
- $f^u() : \{0,1\}^* \rightarrow \mathbb{Z}_u$: pseudo random function (PRF) which maps a bit string of any length to an integer in $\mathbb{Z}_u$.
- A, B, C : IDs of nodes;
- $\mathbb{I}$: Space of IDs. If all IDs are 64 bits long $\mathbb{I} = \mathbb{Z}_{2^{64}}$.
- K_{AB} : pairwise secret between A and B.

II. Key Distribution

A network is a collection of nodes with unique labels (or *identities*). A key distribution scheme for a network consists of a key distribution center (KDC) which provides every node with secrets. Using its secrets a node can a) prove its identity to, and b) establish a private channel with, any node in the network.

A. Key Predistribution Schemes

Key predistribution schemes consist of

1) nodes with unique IDs belonging to the set $\mathbb{I}$, and
2) a key distribution center (KDC).

The KDC chooses a set of P secrets $\mathbb{S}$. A node assigned an ID $A \in \mathbb{I}$ is provided with a key-ring $\mathbb{S}_A$ with k secrets by the KDC. Using its key-ring $\mathbb{S}_A$, node A can compute any $K_{AI} = K_{IA}$ for all $I \in \mathbb{I}$. Similarly, node B can use its key-ring $\mathbb{S}_B$ to compute any $K_{BI} = K_{IB}$ for all $I \in \mathbb{I}$. Thus both A and B can independently compute a common secret $K_{AB} = K_{BA}$. This secret can be used for mutual authentication of nodes and for establishing a private channel between the nodes.

1) Scalable and Nonscalable KPSs: KPSs can be broadly classified into *nonscalable* and *scalable* schemes. For nonscalable schemes the size of the set $\mathbb{I}$ is limited to some value N_{ns}. One example of a nonscalable KPS is the "basic" KPS, where the KDC generates $\binom{N_{ns}}{2}$ pairwise secrets, and provides each node with $k = N_{ns} - 1$ secrets.

For scalable KPSs the set $\mathbb{I}$ is *unlimited* (or network size N is unlimited). Nevertheless, the key-ring size is a *constant* k which is *independent* of the network size N. However, by pooling together the key-rings of (say) n nodes in a set $\mathbb{A} \subset \mathbb{I}$, the attacker may be able to *illegitimately* compute[1] K_{XY} even when $\{X, Y\} \cap \mathbb{A} = \emptyset$. Thus scalable KPSs achieve unrestricted scalability by sacrificing resistance to collusions.

[1]Note that for schemes which are *not* susceptible to collusions such an attacker can only compute secrets of the form K_{XY} if $X \in \mathbb{A}$ *or* $Y \in \mathbb{A}$.

2) Deterministic and Probabilistic KPSs: Scalable KPSs can be classified into n-secure *deterministic* schemes [10] and (n, p)-secure *probabilistic* schemes [13], [11], [12]. For the former, an attacker with access to secrets of n or less nodes cannot determine *any* illegitimate[2] shared secret. However, with access to secrets of *more* than n nodes, the attacker can discover *all* pairwise secrets. Such n-secure deterministic KPSs require $k = \mathbb{O}(n)$ to achieve n-security.

For an (n, p)-secure probabilistic KPS (P-KPS) an attacker with access to secrets of n randomly chosen nodes can compute any illegitimate pairwise secret with a probability p. In other words, such an attacker can compute a *fraction* p of all illegitimate pairwise secrets. For P-KPSs $p(n)$ is a smooth and monotonically increasing function of n. P-KPSs require $k = \mathbb{O}(n \log(1/p))$ to achieve (n, p)-security [12].

B. Assumptions and Network Model

The nodes (with unique IDs) of the network are sensors and proxies manufactured by one or more vendors. Every sensor is equipped with

1) modest hardware for performing symmetric cryptographic computations (block encryption, hashing etc.), and
2) a few tens of kilobytes of storage for keys and software.

The proxy devices are assumed to be comparable in capability to a modern mobile phone / PDA. More specifically, proxy devices can support several Gigabytes of storage using flash storage cards.

The KDC is an entity who

1) verifies the integrity of sensors and proxies;
2) assigns credentials and unique IDs to them, and
3) secrets corresponding to the ID,

before sensors/proxies reach *consumers* who deploy sensor networks.

As an example, a deployer may acquire tens of thousands of sensors and a few tens of proxies and aerially drop thousands of sensors along with one or more proxies in different locations. We shall assume that the specific sensors and proxies acquired by the deployer are randomly chosen from the large number N of devices that have been manufactured.

Once the sensors (along with proxies) are dropped over some remote region, every sensor will be required to establish a pairwise secret with all its "accidental neighbors" who happen to fall close to them, and perhaps a few other sensors / proxies depending on the nature of the application. Note that even while in the field the sensor A may have to establish shared secrets only with

[2]By pooling secrets of other nodes.

a small set of sensors (for example neighbors of A), no information is available regarding the IDs of such nodes at the time the sensor A was provided with secrets (in the factory floor). Thus every sensor should have the ability to establish a secret with any of the N sensors / proxies.

III. Taking Advantage of Proxies

Existing symmetric-only key distribution schemes for large scale sensor networks can be classified into two categories: Kerberos-like schemes, and key predistribution schemes (KPS).

In a scenario where the total number of proxy devices is small, it may be feasible to provide every sensor with a secret it shares with every proxy. Sensors can then use the closest proxy to mediate a session secret with other sensors. The disadvantages of such an approach are three-fold:

1) Proxies are trusted only for relaying measurements: only the sensors are trusted for the measurements reported by them. Thus keys assigned to sensors for purposes of authentication of sensor data (exchanged between sensors) should not be privy to the proxies.
2) An attacker who has compromised a proxy device will be able to impersonate a large number (or even all) sensors. While an attacker who has compromised a proxy device may be able to stop the sensor network deployment (or a part of the deployment) from being useful, it is not desirable for the attacker to be able to impersonate sensors to send misleading information: "no information" is acceptable; deliberately misleading information is not.
3) It is also desirable for sensors to be able to authenticate themselves in scenarios where they cannot gain access to the proxy device.

A second approach is the use of scalable KPSs which provides the ability to establish pairwise secrets without using a mediator. Due to limited storage ability of sensors, such schemes cannot realize high collusion resistance n.

A. Proxy Aided KPS

A third approach proposed in this paper is *proxy-aided* key predistribution schemes (PA-KPS) where proxies aid the sensors in establishing pairwise secrets, without gaining clear access to the secrets of the sensors.

Every sensor stores a unique secret known only to the KDC. Such a secret M_A of sensor A is used for encrypting and/or authenticating A's PA-KPS key-ring. The encrypted key-ring is however stored in a proxy device. The proxies are *not* privy to the secret M_A. Thus sensors employ the proxy as an *untrusted external storage location*. We shall also assume that each proxy has a storage capacity of 32 GB (or $32 \times 2^{32} = 2^{35}$ bytes). Let us also assume that the PA-KPS secrets are 10 bytes long, and thus the total $S = 2^{35}/10$ such values can be stored by a proxy. The question now is *what is a good choice for such a PA-KPS*?

For the desired application model where deployers simply choose a random set of sensors and proxies to be deployed over some region, *every* proxy would need to store the key-rings of *all* N devices. Mandating large N and large key-ring size k will render the storage $S = kN$ required for proxies unacceptable. Clearly, existing KPSs are not very useful here, especially for large N.

We shall now present two KPSs that are good choices for proxy aided KPSs. In both schemes sensors store one or a small number of secrets. The proxies store *public* values (which need not be encrypted). The first scheme is *not* susceptible to collusions and can support reasonably large network sizes N. The second scheme can support *unrestricted* network size N and can offer high levels of collusion resistance.

1) The Leighton-Micali Scheme: Both schemes are variants of an elegant scheme proposed by Leighton and Micali [13] as an alternative to Kerberos. We shall refer to the scheme as LM-KDS (in [13] the authors simply refer to it as "scheme III"). In LM-KDS, the KDC chooses a master key K and a hash function $h()$. An node with ID A is provided with the secret $K_A = h(K, A)$. When A desires to establish a session secret with B, A approaches an *on-line* KDC to receive a public value P_{AB} where

$$P(A, B) = h(K_A, B) \oplus h(K_B, A) = P(B, A). \quad (1)$$

The shared secret between A and B is computed by A as

$$K_{AB} = h(K_A, B) \oplus P(A, B) = h(K_B, A). \quad (2)$$

Note that B can also compute the shared secret $K_{AB} = h(K_B, A)$ directly using its secret K_B (*without* using the public value $P(A, B)$). The value P_{AB} is public as it reveals no information about the secrets of A or B or K_{AB} or K_{BA} to entities other than A or B, and thus need not be kept a secret.

B. A Scheme for Limited N

If node A (with secret K_A) has access to all public values $P(A, i)$ for all $i \in \mathbb{I}$, then LM-KDS becomes a key predistribution scheme, or LM-*KPS*, which facilitates A to compute any secret K_{Ai}. Unlike the basic KPS for which every node is provided with $N - 1$ secrets, in LM-KPS every node needs to store one secret and $N - 1$ public values. Like the basic KPS the LM-KPS is nonscalable. Alternately, LM-KPS can also be used in

conjunction with an untrusted public repository with the ability to store $\binom{N}{2}$ public values. Nodes will simply get the necessary public values from the repository.

The proxies can play this role of untrusted repositories by storing $S = \binom{N}{2}$ public values. For $S = \binom{N}{2} = 2^{35}/10$, we have $N_{max} \approx 83,000$. A node A with neighbors (say) B, C, D can get the required public values $P(A,B), P(A,C), P(A,D)$ from closest proxy and compute K_{AB}, K_{AC} and K_{AD}. On an average every node will be required to get the public values of only half its neighbors as only one of the two nodes at either ends of a link need the public value. It is important to note that access to the public values provides no information about the secrets assigned to sensors.

Before sensors can send a neighbor list to the proxy, sensors will need to identify their neighbors. It is desirable to have some initial neighbor authentication scheme for this purpose to avoid unnecessary overheads[3]. For example, a traditional KPS with low levels of collusion resistance (as only a small key-ring is stored by every sensor) can be used for this purpose.

The sensor A can authenticate its request (consisting of a list of its neighbors) to a proxy R using a secret $h(K_A, R)$ - which the proxy can compute as $h(K_R, A) \oplus P(A,R)$ (as the proxy has access to all public values). For a sensor A with neighbors B, C, D the proxy may send up to three public values in response ($P(A,B), P(A,C)$ and $P(A,D)$). This response will also be authenticated using the shared secret $h(K_A, R)$. While authentication is not strictly required to gain access to the public values, mandating authentication can prevent denial of service attacks. Without this an attacker may be able to send numerous unwarranted requests to the proxy device and thus waste precious bandwidth. Authentication by the proxy can prevent unauthorized modification of the values by an attacker between the proxy and the sensor.

Using the public value $P(A,B)$, A can now compute the pairwise secret $K_{BA} = h(K_A, B) \oplus P(A,B)$, which can be directly computed by B as $K_{BA} = h(K_B, A)$. Only one of the two neighbors require the public value corresponding to the other. Thus, on an average, every node requires half as many public values as the number of its neighbors (to be sent by a proxy).

C. A Scheme for Unlimited N

The nonscalable LM-KPS can be further extended to a *scalable* KPS by using many instances of LM-KPSs in parallel. In the proposed *parallel* Leighton-Micali (PLM) scheme for *unlimited* N, the KDC chooses m parallel schemes, each with M distinct IDs.

The KDC chooses m master secrets, say $K_0 \cdots K_{m-1}$ and computes public $m\binom{M}{2}$ values

$$P_i(j_1, j_2) = h(K_{i,j_1}, j_2) \oplus h(K_{i,j_2}, j_1), j_1 \neq j_2, \quad (3)$$

where $\{j_1, j_2\} \in \mathbb{Z}_M$ and $i \in \mathbb{Z}_m$.

The KDC chooses a public pseudo-random function (PRF) $f^M() : \{0,1\}^* \rightarrow \mathbb{Z}_M$ which maps bit strings of any length to an integer between 0 and $M-1$ (for example, a hash function with $\log_2 M$-bit output). A sensor with ID A is associated with m values $a_0 \cdots a_{m-1}$ where

$$a_i = f^M(A \parallel i), a_i \in \mathbb{Z}_M \forall i \in \mathbb{Z}_m \quad (4)$$

Corresponding to the values $a_0 \cdots a_{m-1}$ the sensor A is provided m secrets $K_{0,a_0} \cdots K_{m-1,a_{m-1}}$ where

$$K_{i,a_i} = h(K_i, a_i), \quad (5)$$

Thus each sensor stores m secrets.

A proxy with ID R is provided secrets m secrets $K_{i,r_i} = h(K_i, r_i = f(R,i)), 0 \leq i \leq m-1$. In addition, *all* $S = m\binom{M}{2}$ public values are also provided to *every* proxy device before they leave the factory floor.

1) Key Establishment: The sensor A relays the list of its neighbors to the proxy R. This list can be authenticated by A using the secret K_{AR} where

$$K_{AR} = h(K_{1,a_1}, r_1) \oplus \cdots \oplus h(K_{m,a_m}, r_m). \quad (6)$$

The proxy device can compute each of the m components of K_{AR} as

$$h(K_{i,a_i}, r_i) = h(K_{i,r_i}, a_i) \oplus P(a_i, r_i), \quad (7)$$

and thus compute the shared secret K_{AR}.

The proxy device relays s values to A - one corresponding to each of A's neighbor. More specifically, corresponding to a neighbor B, A receives a value $Q(A,B)$ from the proxy where

$$Q_{A,B} = P_1(a_1, b_1) \oplus \cdots \oplus P_m(a_m, b_m) \quad (8)$$

where $a_i = f(A,i), b_i = f(B,i)$ for $1 \leq i \leq m$. The shared secret K_{AB} is computed by A as

$$K_{AB} = \{h(K_{1,a_1}, b_1) \oplus \cdots \oplus h(K_{m,a_m})\} \oplus Q_{A,B} \quad (9)$$

Sensor B on the other hand can simply compute K_{AB} as

$$K_{AB} = h(K_{1,b_1}, a_1) \oplus \cdots \oplus h(K_{m,b_m}, a_m). \quad (10)$$

[3]For example, an attacker could cause nodes to send lists containing nonexistent sensors. Only after the response from the proxy will sensors realize that such nodes do not exist.

D. Collusion Resistance

It is assumed that the attacker has access to *all* $S = m\binom{M}{2}$ public values. In addition the attacker has the secrets of n sensors. An attacker who has secrets of a sensor C may be able to compute $h(K_{j,a_j}, b_j)$ for some j *if*

$$f^M(C \parallel j) = c_j \in \{a_j, b_j\}. \tag{11}$$

More specifically, if $c_j = a_j$, C has access to the secret K_{j,a_j} and can trivially compute $h(K_{j,a_j}, b_j)$. If $c_j = b_j$, C has access to the secret K_{j,b_j} he can compute $h(K_{j,b_j}, a_j) \oplus P_j(a_j, b_j) = h(K_{j,a_j}, b_j)$. As the PRF $f^M()$ produces uniformly distributed random numbers between 0 and $M-1$ the probability of such an occurrence ($f^M(C \parallel i) = c_i \in \{a_i, b_i\}$) is

$$\gamma = \frac{2(M-1)}{M^2} \approx 2/M, \tag{12}$$

for large M. The probability that a specific $h(K_{i,a_i}, b_i)$ or $h(K_{i,b_i}, a_i)$ *cannot* be computed by a an attacker with secrets of n nodes is thus $(1-\gamma)^n \approx \mathrm{e}^{-\gamma \mathrm{n}}$. In order to compute a pairwise secret K_{AB} the attacker has to compute $h(K_{i,a_i}, b_i)$ or $h(K_{i,b_i}, a_i)$ for *all* $0 \leq i \leq m-1$. The probability $p(n)$ that the attacker *can* compute K_{AB} is then

$$p(n) \approx (1 - \mathrm{e}^{-2\mathrm{n}/\mathrm{M}})^{\mathrm{m}} \tag{13}$$

1) Choice of Parameters m and M: From Eq 13 it can be readily seen that by choosing $m \propto \log(1/p)$ and $M \propto n$ we can achieve any desired $p(n)$. Thus the storage complexity for the proxy for $S = m\binom{M}{2}$ secrets is $\mathbb{O}(n^2 \log(1/p))$.

For $S = m\binom{M}{2} < 2^{35}/10$, some feasible choices of parameters (m, M) are then $(m = 2, M = 58617)$, $(m = 3, M = 47861)$, $(4, 41449)$, $(5, 37073)$, $(6, 33843)$, $(4, 41449)$, $(7, 31332)$, $(8, 29309)$, $(12, 23930)$, $(16, 20724)$, $(24, 16921)$, $(32, 14654)$, $(48, 11965)$, $(m = 64, M = 10362)$, $(m = 80, M = 9268)$, $(m = 96, M = 8461)$, etc.

Table 1 depicts the number of sensors n_j to be compromised by an attacker to compromise a fraction $p_j = p(n_j)$ of all link secrets for various such values of m, M. For example, for $m = 48$ and $M = 11965$, the attacker has to compromise $n_1 = 3410$ sensors for illegitimately compromising one in a trillion links; the attacker has to compromise secrets from $n_2 = 4580, n_3 = 6440, n_4 = 9990, n_5 = 13830$ and $n_6 = 19130$ sensors for compromising one in a billion, million, thousand, 512, and 4 links respectively (or $p_j = p(n_j) = \{2^{-40}, 2^{-30}, 2^{-20}, 2^{-10}, 2^{-5}, 2^{-2}\}$ respectively).

Note that for small m (and larger M as $S = m\binom{M}{2}$), $p(n)$ degrades more gracefully. The choice of larger m (smaller M) performs better for smaller n but degrades faster. This is not surprising considering Eq 13, where for some desired $p(n)$, $M \propto n$.

TABLE I

COLLUSION RESISTANCE

PLM: *unlimited* network size N

m	M	n_1	n_2	n_3	n_4	n_5	n_6
12	23930	430	1020	2500	6820	12880	22420
16	20724	880	1720	3490	7940	13620	22180
24	16921	1770	2850	4820	9100	14130	21370
32	14654	2470	3640	5610	9630	14160	20540
48	11965	3410	4580	6440	9990	13830	19130
64	10362	3970	5090	6810	10010	13400	18030
80	9268	4320	5380	6990	9920	12990	17150
96	8461	4550	5560	7060	9780	12610	16420

KSSC for network size $N = 85,000$

m'	M'	n_1	n_2	n_3	n_4	n_5	n_6
20	2021	1400	1820	2480	3700	5030	6830

Subset allocation Scheme for network size $N = 85,000$

t	k	n_1	n_2	n_3	n_4	n_5	n_6
536	40424	535	690	905	1278	1650	2140

E. Other Probabilistic KPSs

All KPSs are essentially trade-offs between security and complexity. The complexity of KPSs however has three facets [3]: i) storage complexity for secrets; ii) computational complexity (for computing pairwise secrets); and iii) memory-fetch complexity (the number of secrets that need to fetched from the key ring and used for computing any pairwise secret). For n-secure deterministic schemes *all* three facets of complexity are $\mathbb{O}(n)$.

For (n, p)-secure probabilistic KPSs (P-KPS) the storage complexity is $k = \mathbb{O}(n \log(1/p))$. The computational complexity and memory-fetch complexity are $\mathbb{O}(\log(1/p))$, which is *independent* of n [3].

For the well known schemes based on subset allocation, defined by parameters (k, ξ) [11], [12] the KDC chooses $P = k/\xi$ secrets and provides k secrets to every node. Any two node will share $k\xi$ secrets on an average. For such schemes

$$p(n) = (1 - \xi(1-\xi)^n)^k. \tag{14}$$

Realizing a (n, p)-secure scheme calls for $k \geq 2.72n \log(1/p)$, and for the minimum value of k we need to choose and $\xi \approx 1/n$ [12]. The KDC chooses $2.72 \log(1/p) n^2$ secrets and provides every node with $2.72 \log(1/p) n$ secrets. Any two nodes will share $2.72 \log(1/p)$ secrets on an average.

For a more efficient P-KPS proposed recently - the key subset and symmetric certificates (KSSC) scheme [14], defined by two parameters m' and M',

$$p(n) \approx \left(1 - \mathrm{e}^{-\mathrm{n}/\mathrm{M}'}\right)^{2m'}. \tag{15}$$

The KDC chooses $M'm'$ secrets and provides each node with m' secrets and $m'M'$ symmetric certificates (which are also treated by nodes as secrets). For (n,p)-secure KSSC each node needs to store $m' = \log_2(1/p)$ secrets and $k = m'M' \approx 1.04n\log(1/p)$ symmetric certificates.

1) Advantages of P-KPSs: Some of the well recognized advantages of P-KPSs (over deterministic schemes) are i) their low hardware complexity: only a block-cipher or hashing is required; finite-field arithmetic is not necessary; and ii) their security degrades gracefully with increasing number of compromised nodes; and iii) their ability to employ multi-path diversity to improve their security [7].

Two other compelling advantages come from their i) low memory-fetch complexity and ii) low complexity of operations performed with secrets.

For deterministic KPSs where all k secrets are used for computing any pairwise secret, we cannot afford to store the sensor's secrets in a proxy and "fetch them when necessary." P-KPSs can however utilize external storage and fetch the small number of secrets required when required.

Secondly, as deployments may be unmonitored, it is necessary to consider the possibility of attackers who may desire to impersonate genuine sensors for purposes of sending misleading information. Thus, not withstanding their low cost, sensors will need to be tamper-responsive, and zeroise secrets under suspicions of intrusions. Due to the modest complexity of operations (for P-KPSs) performed with secrets we can afford to employ a low-complexity low-power chip to perform such computations. Such chips can be afforded better protection at low cost [3].

F. PLM vs Other P-KPSs

Even while P-KPSs have compelling advantages over deterministic KPSs, existing P-KPSs are still not good choices for PA-KPSs. The reason for this is that we desire *any sensor to be able to employ any proxy*, as it may be inconvenient to bind specific sensors to specific proxies during the time the keys were provided to the sensors and proxies. The implication then is that *every proxy should store the values assigned to every sensor*. Obviously for large N we cannot afford to assign a large number of PA-KPS values (k) to every sensor as we require $kN \leq S$.

For a subset allocation scheme with parameters (k,ξ) the KDC chooses $P = k/\xi$ independent secrets. If the proxy could just store the P secrets the acheivable collusion resistance will be independent of the network size N. Unfortunately, though one of the P secrets (say K_i) may be assigned to $N\xi$ nodes (as each node gets any secret with a probability $0 \leq \xi \leq 1$), the proxy needs to store $N\xi$ copies of the same secret - each copy encrypted with a secret privy only to one node (as the proxies should not gain clear access to the secrets assigned to sensors).

Note that PLM can also be used a traditional P-KPS if every sensor also stores $k = mM$ public values (in addition to m secrets). The storage complexity is thus $\mathbb{O}(n\log(1/p))$. For computing K_{AB}, A has i) compute $b_i = f(B,i), 0 \leq i \leq m-1$, ii) fetch m public values $P(a_i,b_i)$, and iii) compute m hashes. Thus the computational complexity and memory-fetch complexity are independent of n.

However the primary advantage of PLM over other P-KPSs (like subset allocation schemes and KSSC) stems from the fact that for a network size of N the proxy need not store $Nk = NmM$ public values. The proxies need to store only $m\binom{M}{2}$ values - which is *independent* of N. As public values need not be encrypted, one public value can be used by multiple nodes. For example, a public value $P_i(a,b)$ can be used by any node for which the i^{th}short ID is a or b. On an average, a specific public value $P_i(a,b)$ can be useful for $\frac{2N}{mM}$ nodes.

Thus for a give storage limitation S, for traditional P-KPSs $n \propto 1/N$. For PLM $n \propto \sqrt{S}$, which is independent of N. Or

$$S = \begin{cases} kN \propto nN & \text{Traditional P-KPSs} \\ m\binom{M}{2} \propto n^2 & \text{PLM} \end{cases} \tag{16}$$

In the rest of this section we will show why for *any* storage limitation S and for *any* network size N the best choice is *either* the nonscalable LM-KPS or PLM. To demonstrate this we shall show that this is indeed the case for a fixed S, and then argue why this holds for *any* S.

First let us consider the case where the storage capacity for the proxy is $S = 2^{35}/10$. For the assumed storage capacity, $S = 2^{35}/10$ values for each proxy, as long as the network size is less than $N_{ns} = 85,000$, *we have no reason to choose a collusion susceptible KPSs*. Similarly for very large N PLM is obviously a good choice. For unlimited N existing P-KPSs are not useful as the acheivable collusion resistance $n \propto 1/N$.

The question now is for $N > N_{ns} + \epsilon$ (which rules out the use of LM-KPS), would a traditional P-KPS perform better than PLM? If the performance of existing P-KPSs (subset allocation schemes and KSSC) is inferior to PLM even for $N = N_{ns}$, obviously they will be inferior for larger N as $n \propto 1/N$ (for a given S).

For a subset allocation scheme for a network size of $N = 85,000$ and $S = kN = 2^{35}/10$ we have $k \leq 40424$. For KSSC for a network size of $N = 85,000$ we can choose $m' = 20$ and $M' = 2021$. The storage corresponding to each sensor is $k = m'M' = 40420$.

The achievable $p(n)$ for a subset allocation scheme with $k = 40424, t = 536$ and a KSSC scheme with $m' = 20$ and $M' = 2021$, are also indicated in Table 1 for $N = 85,000$.

Clearly, the performance of PLM is superior to KSSC and subset allocation scheme for *this particular choice* of proxy storage capability $S = 2^{35}/10$. However, it is easy to see that this is true for *any* proxy storage capability.

If the storage capability of proxies increase by a factor 100

1) we can realize a ten-fold improvement in collusion resistance n of PLM (as $S \propto n^2$);
2) a hundred fold increase in S also implies a ten-fold increase in the network size N_{ns} that can be supported by the nonscalable LM-KPS (N_{ns} increases to 850,000);
3) as there is no reason to consider a scalable KPSs for $N < 850,000$, a 100 fold increase in $S = kN$ (for other P-KPSs) implies
   a) a ten-fold increase in N, and
   b) a ten-fold increase in k.

   Thus the collusion resistance improves by a factor 10 (same as PLM).

In other words, for *any* storage capability of the proxies, and for *any* network size, the best choice for PA-KPS is

1) the nonscalable LM-KPS (if the network size N is such that $S \geq \binom{N}{2}$, or
2) PLM, for large / unlimited network size.

IV. Conclusions

We proposed a novel key distribution scheme for sensor network applications involving deployments of a large number of resource constrained sensors along with few more capable proxy devices. We argued that current approaches for establishing pairwise secrets between sensors either i) require sensors to trust the proxy device with their secrets or ii) settle for low collusion resistance due to the limited resources of sensors. In this paper we presented a third approach where proxies serve as untrusted storage resources for the sensors. The proxies are not trusted with the secrets of the sensors. The proposed two schemes which can take good advantage of the storage resources of proxies.

The first scheme, LM-KPS is not susceptible to collusions, and can achieve reasonable large network sizes (many tens of thousands). The second scheme can realize *unlimited* network sizes while providing high levels of collusion resistance.

At the crux of both schemes is that the proxies need to store only public values. For the nonscalable scheme each public value can be used by a pair of nodes. For the scalable PLM a public value $P_i(j_1, j_2)$ can be used by any node for which the i^{th}short ID is j_1 or j_2. In both schemes a node with r neighbors will need to fetch $r/2$ public values from the closest proxy in order to establish a secret with all neighbors.

References

[1] C-Y Chong, Kumar, S.P., "Sensor networks: Evolution, opportunities, and challenges," Proc IEEE, August 2003.

[2] T. He, C. Huang, B. Blum, J. Stankovic, T. Abdelzaher. "Range-Free Localization Schemes in Large Scale Sensor Networks," Mobile Computing and Networking (MobiCom2003), 2003.

[3] M. Ramkumar, "Trustworthy Computing Under Resource Constraints With the DOWN Policy," IEEE Transactions on Secure and Dependable Computing, Jan 2008.

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

[5] A. Perrig, R. Szewczyk, J.D. Tygar, D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks* **8** 521–534, 2002.

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

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

[8] Di Pietro, R., Mancini, L. V., and Mei, A., "Random key assignment for secure wireless sensor networks," Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), Fairfax, VA, USA, 62–71, 2003.

[9] 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.

[10] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Advances in Cryptology: Proc. of Eurocrypt 84*, Lecture Notes in Computer Science, **209**, Springer-Verlag, Berlin, pp. 335-338, 1984.

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

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

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

[14] M. Ramkumar, "Efficient Key Distribution Schemes for Large Scale Mobile Computing Applications," Cryptology ePrint Archive, 2008/332, http://eprint.iacr.org/2008/332.pdf.