On the Complexity of Probabilistic Key Predistribution Schemes

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*— Collusion susceptible key predistribution schemes (KPS) are trade-offs between complexity and security. Measures of complexity include storage, computation and bandwidth overhead; a measure of security is their collusion resistance. Probabilistic KPSs (P-KPS) have some unique advantages in application scenarios where hardware assisted protection of secrets is mandatory. To facilitate a quantitative comparison of different facets of complexity associated with KPSs we investigate two generic device models suitable for a wide range of application scenarios where hardware assisted protection of secrets is mandatory. We propose a novel P-KPS, the subset keys and identity tickets (SKIT) scheme, which demands substantially lower overheads compared to other P-KPSs for such application scenarios.**

I. Introduction

A communication network is a collection of nodes with unique labels and credentials. Securing interactions between two nodes with labels A and B require mechanisms which permit the nodes to verify each others labels and credentials, and establish a pairwise secret K_{AB}. The secret K_{AB} can be leveraged for privacy and mutual authentication of messages exchanged between the nodes.

Any cryptographic security association is at most only as strong as the mechanism used for protecting secrets assigned to nodes. In more conventional networks the secrets assigned to nodes (computers) are protected by human beings, perhaps by controlling physical access to the computers. Unfortunately, for a wide variety of emerging application scenarios nodes may be computers deployed in an unattended manner. In such cases there is a need for *hardware assisted protection of secrets* assigned to devices. In such scenarios, there are good reasons to limit the complexity of the trustworthy hardware security modules (HSMs) which protect and perform cryptographic computations using such secrets.

Scalable key predistribution schemes (KPS) which employ only symmetric cryptography are well suited for securing interactions between nodes equipped with HSMs. Scalable KPSs require every node to store a limited set of k secrets, but can nevertheless support *unlimited* network sizes N. Irrespective of the network size N, an n-secure KPS can resist collusions of up to n nodes pooling their secrets together. More generally, for an (n, p)-secure probabilistic KPS (P-KPS), an attacker pooling together secrets of n nodes can determine any pairwise secret with a probability p.

There are two reasons which render scalable KPSs well suited for use in conjunction with HSMs. Firstly, due to the low computational overhead, the use of KPSs can permit the use of HSMs with low complexity (in general, lower the complexity of HSMs, higher their reliability). Secondly, that it may be impractical for attackers to expose secrets protected by *many* HSMs can render the issue of susceptibility to collusions less of a concern.

A. Contributions

Traditionally the measure of the “efficiency” of any KPS has been regarded as n/k, where k is the number of keys to be stored by each node to realize a n-secure KPS. This *was* a reasonable measure, as for most of the early *deterministic* KPSs in the literature, storage and computational overheads were linear in k (and n, as for most KPSs $k \propto n$). This is however not true for P-KPSs.

For the intended purpose we desire KPSs which can provide high collusion resistance while keeping the HSM complexity low. We argue that P-KPSs in general are better suited than deterministic KPSs for the intended application. To facilitate a detailed analysis of the overhead associated with probabilistic KPSs we take an in-depth look at a generic class of applications where there is a need for hardware assisted protection of secrets. We provide two generic device models suitable for a wide range of emerging application scenarios like sensor and ad hoc networks, and more generally, large scale networks of resource limited ubiquitous computers.

This paper introduces a novel P-KPS, the *subset keys and identity tickets* (SKIT) scheme. The generic device models are used to demonstrate that SKIT is *substantially* more efficient compared to better known P-KPSs based on subset intersection (SI).

The rest of this paper is organized as follows. Section II provides an overview of key predistribution schemes. The novel SKIT scheme is described in Section III. Section IV outlines the generic device models and provides an in-depth comparison of complexity of various KPSs. Section V argues why deterministic KPSs are ill-suited for the intended application scenario. Several advantages of probabilistic KPSs in general, and SKIT in particular are enumerated. Section V-A discusses some related work. Conclusions are offered in Section V-B.

II. Key Pre-Distribution Schemes

In key pre-distribution schemes (KPS) a key distribution center (KDC) chooses a set of P secrets $\mathbb{S}$. Every node is assigned a unique identity from a set $\mathbb{I}$. A node with identity $A \in \mathbb{I}$ is issued a key-ring $\mathbb{S}_A$ with k secrets. Using its key-ring, A can compute $K_{Ai} = K_{iA}$ for *all* $i \in \mathbb{I}$. Likewise, node B can use its key-ring $\mathbb{S}_B$ to compute $K_{Bi} = K_{iB}$ for all $i \in \mathbb{I}$. Thus, both A and B can independently compute a common secret $K_{AB} = K_{BA}$.

KPSs can be broadly classified into non-scalable and scalable schemes. For non-scalable KPSs the size of the set $\mathbb{I}$ is limited to some value M - or $\mathbb{I} \equiv \mathbb{Z}_M = \{0, 1, 2, \ldots, M-1\}$. For the non-scalable "basic" KPS, for a network size of M, the KDC generates $\binom{M}{2}$ pairwise secrets and provides each node with a key-ring with $k = M - 1$ secrets.

For scalable KPSs the set $\mathbb{I}$ is *unlimited.* Nevertheless, using its restricted set of k keys, a node can *compute* a practically unlimited number ($| \mathbb{I} |$) of pairwise secrets. However, unlike non-scalable KPSs, scalable KPSs are susceptible to collusions. An entity with access to the secrets of n nodes in the set $\mathbb{A}$ may be able to *pool* the key-rings together to *illegitimately* compute K_{uv} even when $u, v \notin \mathbb{A}$.

Scalable KPSs can be classified into deterministic n-secure schemes and probabilistic (n, p)-secure schemes. For the former, an attacker with access to secrets of n or less nodes cannot compute any illegitimate secret. However, with access to secrets of more than n nodes the attacker can compute *all* secrets. For (n, p)-secure schemes an attacker with access to secrets of n randomly chosen nodes can compute a fraction $p(n)$ of all illegitimate pairwise secrets. As long as p is sufficiently small (for example, 2^{-64}) it is *computationally infeasible* for an attacker to even determine *which* pairwise secrets can be computed using the secrets of n nodes.

A. A Deterministic KPS: Blom's SKGS

That it is possible to realize unlimited network scales by sacrificing resistance to collusions was first recognized by Blom [2]. In [2] and [3] Blom proposed schemes which rely on finite field arithmetic to generate the key ring of any node from a set of keys chosen by the KDC. The scheme based on symmetric polynomials [2] required each node to store about $k \approx 2n$ keys to achieve n-security. The symmetric key generation scheme (SKGS) [3] which requires every node to store $k = n + 1$ keys remains the most storage efficient scheme.

To realize an n-secure Blom's SKGS over a finite-field $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ the KDC chooses i) a generator $\alpha \in \mathbb{Z}_q$; and ii) a $(n+1) \times (n+1)$ symmetric matrix $\mathbf{D}$ with $P = \binom{n+1}{2}$ values chosen randomly from $\mathbb{Z}_q$. The P values are the KDC secrets $\mathbb{S}$.

To provide keys to a node with identity $A \in \mathbb{Z}_q$, the KDC computes $\mathbf{g}_A = [g_A(0), g_A(1), \ldots, g_A(n)]^T$, where $g_A(i) = \alpha^{iA}$, and provides a set of k secrets $\mathbb{S}_A = \mathbf{D}\mathbf{g}_A = \mathbf{d}^A = [s_0^A, s_1^A, \ldots s_n^A]^T$.

Nodes A and B (with secrets $\mathbf{d}^A$ and $\mathbf{d}^B = [s_0^B, \cdots s_n^B]$ respectively) can compute $K_{AB} = (\mathbf{d^A})^\mathbf{T}\mathbf{g_B} = (\mathbf{d^B})^\mathbf{T}\mathbf{g_A}$ (as $\mathbf{D}$ is a symmetric matrix) as

$$K_{AB} = \begin{cases} \sum_{i=0}^{n} s_i^A \alpha^{iB} \bmod q & \text{by } A \\ \sum_{i=0}^{n} s_i^B \alpha^{iA} \bmod q & \text{by } B \end{cases} \quad (1)$$

B. Probabilistic Subset Intersection Schemes

One major limitation of schemes that require finite-field multiplications is the more complex hardware required for this purpose. This was the motivation for Gong and Wheeler [4] to propose a key predistribution scheme which can employ any symmetric block-cipher or hash function as a building block. In the matrix [4] scheme for a network size of N, each node will require $n\sqrt{N}$ keys to resist collusion of up to n nodes. The matrix scheme was the first in the category of *subset* intersection schemes where the KDC chooses a set of P secrets and each node receives a subset of $k < P$ secrets, and all keys common to two nodes are used to compute pairwise secrets.

Later (in 1995), influenced by the seminal work of Erdos et al [6], Mitchell and Piper [5] investigated strategies for subset allocation for a given network size N, subject to the constraint that "the union of keys belonging to no n nodes should cover the intersection of keys of any two nodes." The scheme in [5] required every node to receive $k = \mathbb{O}(n \log N)$ secrets.

Unfortunately, the schemes in [4] and [5] were not truly scalable as the key allocation strategy is tied to the *total* number of nodes N. In other words, for such schemes it is not possible to *add* new nodes to the network. In 1995 Dyer et al [7] argued that complex allocation strategies employed by Mitchell and Piper [5] can be replaced by random or pseudo-random allocation of subsets. More importantly, as the allocation strategy is not tied to the total number of nodes, the approach proposed by Dyer et al [7] results in a truly scalable scheme (like the earlier deterministic KPSs) where nodes can be added to the network at any time, and there is no limit on N - the total number of nodes.

In 1999 [8] the idea of random subset allocation was applied for constructing a broadcast authentication scheme with probabilistic assurances. More recently, since [9] in 2002, the concept of random subset allocation has been applied by various researchers for securing sensor networks [9] - [14].

While different subset intersection schemes differ in some details regarding *how* the subsets are generated, their (n,p)-security does not depend on the specific distribution strategy. For an (n,p)-secure subset intersection (SI) scheme [7] with parameters $k = n\mathrm{e}\log(1/\mathrm{p})$ and $t = n+1$ [15], the KDC chooses $P = kt$ secrets $\mathbb{S} = \{K_{i,j}\}$, $0 \le i \le k-1, 0 \le j \le t-1$. Each node is provided k secrets.

To provide keys to node A the KDC evaluates k public pseudo-random functions (PRFs) $a_i = f_t(A,i), 0 \le i \le k-1$ where $0 \le a_i \le t-1, \forall i$. The values a_i are the indices of the k secrets provided to A. Or $\mathbb{S}_A = \{K_{0,a_0}, \ldots, K_{k-1,a_{k-1}}\}$.

To compute K_{AB} both A and B evaluate $f_t(A,i)$ and $f_t(B,i)$ for $0 \le i \le k-1$ and determine the indices $i \in \mathbb{Z}_k$ for which $a_i = b_i$. On an *average*, this will occur for $k_f = k/t \approx \mathrm{e}\log(1/\mathrm{p})$ indices. The k_f secrets corresponding to the common indices are XOR-ed together to derive K_{AB}.

III. SKIT: Subset Keys and Identity Tickets

An "identity ticket" is derived as a one way function of a label and a secret. For example, $h(K,A)$ is an identity ticket corresponding to a key K and a label A. While identity tickets are conceptually similar to hashed message authentication codes (HMACs), unlike HMACs, identity tickets are treated as secrets. The identity ticket $h(K,A)$ is privy only to the entity with identity A and entities who have access to the secret K (and can thus *compute* the ticket).

The subset key and identity tickets (SKIT) scheme is defined by two parameters (m,M). The KDC chooses a pre-image resistant hash function $h()$ and a PRF $f_M()$. The KDC chooses a set of $k = mM$ secrets $\mathbb{S} = \{K_{i,j}\}$, $0 \le i \le m-1$, $0 \le j \le M-1$. For a node A the KDC computes $a_i = f_M(A,i), 0 \le i \le m-1$. A is assigned i) m secrets $\mathbb{S}_A$ and ii) mM ITs $\mathbb{T}_A$ where

$$\begin{array}{lcl} \mathbb{S}_A & = & \{K(0,a_0), K(1,a_1), \ldots, K(m-1,a_{m-1})\} \\ \mathbb{T}_A & = & \{T_A(i,j)\}, 0 \le i \le m-1, 0 \le j \le M-1 \\ \text{where} & & T_A(i,j) = h(K_{i,j}, A). \end{array}$$

Two nodes A and B can compute $2m$ common tickets of the form $T_A(i,b_i)$ for $0 \le i \le m-1$ and $T_B(i,a_i)$ for $0 \le i \le m-1$. More specifically,

1) A evaluates PRF $\{b_i = f_M(B,i)\}$ = to choose m of its mM tickets: $T_A(i,b_i) \in \mathbb{T}_A, 0 \le i \le m-1$; A computes m tickets $T_B(i,a_i) \in \mathbb{T}_B$ using its secrets $\mathbb{S}_A = \{K_{i,a_i}\}$ (as $T_B(i,a_i) = h(K_{i,a_i}, B)$).

2) B evaluates PRF $\{a_i = f_M(A,i)\}$ = to determine the indices of m tickets $T_B(i,a_i)$; B employs $\mathbb{S}_B$ to compute m tickets $T_A(i,b_i)$.

All $2m$ tickets are used to derive the pairwise secret K_{AB}, by XOR-ing them together:

$$\begin{array}{rcl} K_{AB} & = & \{T_A(0,b_0) \oplus \cdots \oplus T_A(m-1,b_{m-1}\} \\ & \oplus & \{T_B(0,a_0) \oplus \cdots \oplus T_B(m-1,a_{m-1}\} \end{array} \quad (2)$$

A. (n,p)-*Security*

The tickets assigned to a node do not reveal any information about the secrets or the tickets of *other* nodes (as $h()$ is pre-image resistant). However, an attacker who has access to secrets of n nodes may be able to pool their secrets (not tickets) together to determine all m secrets of A ($\mathbb{S}_A = \{K_{0,a_0} \cdots K_{m-1,a_{m-1}}\}$) *and* all m secrets of B ($\mathbb{S}_B = \{K_{0,b_0} \cdots K_{m-1,b_{m-1}}\}$), from which the required $2m$ tickets, and hence K_{AB}, can be *computed*.

A *specific* node C in the attacker pool is also assigned the secret $K(i,a_i)$ *if* the $f_M(A,i) = a_i = f_M(C,i) = c_i$. The probability of such an event is $1/M$. The probability that a particular $K(i,a_i)$ is *not* assigned to *any* of the n nodes in the attacker's pool is then

$$\epsilon = (1 - 1/M)^n \approx \mathrm{e}^{-\mathrm{n}/\mathrm{M}} \quad (3)$$

(for large M). Thus, the probability that the attackers pool of secrets *can* be used to compute K_{AB} (or *all* $2m$ tickets can be computed by the attacker) is

$$p(n) = (1-\epsilon)^{2m} \approx \left(1 - \mathrm{e}^{-\mathrm{n}/\mathrm{M}}\right)^{2m}. \quad (4)$$

We can rewrite Eq (4) as

$$k = mM = \frac{n\log(1/p)}{-2\theta\log(1-\mathrm{e}^{-\theta})}, \text{ where } \theta = \frac{n}{M}. \quad (5)$$

By choosing $\theta = n/M = \log(2)$ we can maximize $-\theta\log(1-\mathrm{e}^{-\theta})$, and thus minimize $k = mM$. If we desire $p(n^*) \le p^*$ (or a (n^*,p^*)-secure SKIT scheme) the choice of parameters m and M that minimize k (the number of ITs that need to be stored by every node) are

$$\left.\begin{array}{rl} m = & \frac{\log(1/p^*)}{2\log 2} \\ M = & \frac{n^*}{\log 2} \end{array}\right\} \Rightarrow k = \frac{n^*\log(1/p^*)}{2\log^2 2}. \quad (6)$$

For SKIT with parameters $m = 32$ and $M = 2^{15}$ minimizes $k = mM$ to acheive $p(n^* = 22{,}500) < p^* = 2^{-64}$). Each node stores $m = 32$ secrets and $mM = 2^{20}$ (a million) tickets. For 80-bit tickets the storage required is 10 MB. An attacker who has access to secrets of $n^* = 22{,}500$ nodes can determine only a fraction $p^* = 2^{-64}$ of illegitimate pairwise secrets. An attacker who has exposed secrets from $42{,}200 > n^*$ nodes can expose one in a billion pairwise secrets (or $p(42200) \approx 2^{-30} > p^*$). Doubling M to 2^{16} (for the same $m = 32$) will result in a scheme for which $p(45000) \approx 2^{-64}$ (or n^* is doubled). The storage requirement mM is doubled to 20 MB. If we desire a SKIT scheme with $p(100000) \approx 2^{-64}$, about 44 MB of storage is required for every node.

Note that the number of PRFs and hash computations that need to be evaluated by any node (both m), are *independent* of the desired collusion resistance n. Only storage for tickets (mM) limits the achievable collusion resistance. It is for this reason that we attempt to minimize mM to achieve a desired (n, p)-secure SKIT scheme.

IV. Complexity of KPSs

For purposes of comparison of different KPSs, we shall compare the overhead required for an n^*-secure Blom's SKGS, (n^*, p^*)-secure SI, and SKIT schemes, designed to meet the requirement $n^* = 2^{16}$ and $p^* = 2^{-64}$. The SKGS scheme will require every node to store $k = n + 1 = 65,537$ keys. Realizing a $(n = 2^{16}, p = 2^{-64})$-secure RPS calls for $k \approx 7,903,000$ and $t \approx 2^{16}$. The $(n = 2^{16}, p = 2^{-64})$-secure SKIT scheme calls for $m = 32$ and $M = 94,548$. The SI scheme will require every node to store close to 8 million secrets. For SKIT every node will need to store $m = 32$ secrets and a little over 3 million (mM) identity tickets.

From the perspective of *storage complexity* $\mathcal{S}$, SKGS is more efficient. Among the two P-KPSs, SKIT is more than twice as efficient as SI scheme. However, as we shall see in the reminder of this section, the value k alone does not provide the "full picture."

Facilitating any cryptographic authentication scheme mandates some overhead. In general, the overhead may take many forms like computation, bandwidth and storage. The costs associated with different types of overhead will obviously depend on the intended purpose of the network and the nature of devices that constitute the network.

A. Application Model

In conventional networks "nodes" are typically personal computers. Secrets assigned to personal computers are protected by a person, usually by restricting physical access to the computer. In contrast, in emerging ubiquitous computing application scenarios where most devices may be deployed in an unattended manner, the secrets assigned to a computer will require hardware assisted protection. The choice of cryptographic authentication strategies for such networks should thus consider *additional* constraints imposed by the need to employ tamper-resistant hardware security modules (HSM) in devices to *protect and perform computations with secrets*.

Some such computers may be sensors entrusted with the task of monitoring and reporting environmental parameters like temperature, humidity, and levels of atmospheric pollutants; various unattended sensors may be deployed at different locations ranging from rain forests to street intersections in cities, to homes. Some devices may be entrusted with the task of simply routing information between computers, by adhering to some ad hoc routing protocol.

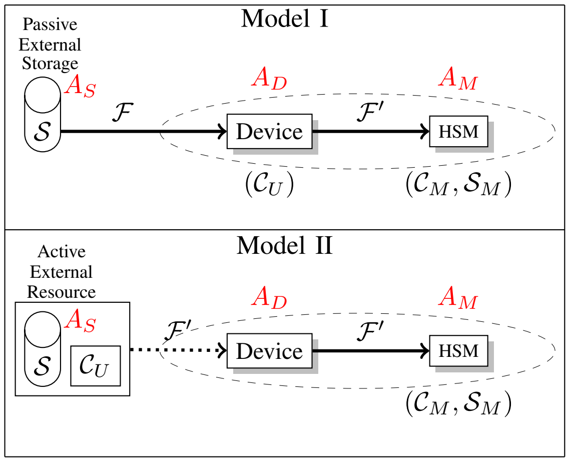

Fig. 1. Generic device model for emerging applications. A node A includes the device A_D and a HSM A_M housed in the device. Devices have access to storage where the key ring is stored - either *passive* pluggable storage (Model I) or *active* external resource, accessed over a network (Model II). Different facets of complexity ($\mathcal{S}$, $\mathcal{C}_U$, $\mathcal{F}$, $\mathcal{F}'$, $\mathcal{C}_M$, and $\mathcal{S}_M$) borne by the three components of a node are also indicated.

In large-scale networks where proactive mechanisms to protect secrets are *mandatory in any case*, KPSs do merit consideration as an alternative to more conventional schemes based on asymmetric cryptography. This is especially true with assurances that i) it is expensive for attackers to tamper with such HSMs and expose secrets, and that ii) the collusion resistance n can be increased to very large extents (or an attacker has to break *many* "unbreakable" HSMs).

B. Device Model

A pictorial view of a node, consisting of a device and an associated HSM, is shown in Figure 1. More specifically, a node with KPS identity A consists of a device A_D and a HSM A_M. The device A_D also has access to a storage device A_S. In the figure we distinguish between two device models:

1) Model I with *passive* external device A_S, and
2) Model II with an *active* external resource A_S.

Model I includes a wide variety of devices that can support pluggable flash storage A_S. For example, A_D can be a mobile phone, and A_S is a flash card plugged into the mobile phone. In this case the HSM could be the SIM[1] card (or a chip inside the SIM card).

In Model II the active external resource is generally more capable than the device A_D. For example, i) A_D can be a sensor device in some home equipment (like a coffee-maker or microwave) which can access a desktop

[1]Subscriber Identity Module.

computer A_S over a blue-tooth link; ii) A_D can be a sensor in a conventional sensor network, and A_S can be a proxy (shared by many such sensors in the network); iii) the device A_D can be a cardiac sensor which has infrared/blue-tooth access to a mobile phone A_S carried by the person. More generally, the "link" between A_D and the active external resource A_S can even span over some wide area network like the Internet.

In both Models I and II the external device (active or passive) is not trusted by the HSM.

In scenarios where it is necessary to employ tamper-resistant hardware security modules (HSM) to protect secrets assigned to devices, there are good reasons to *deliberately* limit the complexity and the power consumed by such HSMs. Low complexity HSMs can be more readily verified and certified for integrity. Low-power modules can be extended unconstrained shielding from intrusions, as heat-dissipation will not be an issue. Consequently, low-complexity-low-power HSMs can be realized at low cost [16] without sacrificing their integrity. Furthermore, in many scenarios the devices may themselves be resource limited, and thus unable to house power-hungry HSMs.

While P-KPSs require more storage compared to deterministic KPSs, this is not a concern for the HSMs. The secrets to be protected by HSM A_M can be encrypted using a small number of secrets stored inside the HSM A_M, and the encrypted secrets can be stored outside the HSM in A_S. Furthermore, it is not necessary for the HSM to compute the PRFs, as long as the HSM is able to *verify* the PRF computations. Thus, just as storage can be off-loaded to resources outside the HSM, so can PRF computations.

To compute any pairwise secret like K_{AB} the HSM A_M will need to perform some computations - some for verifying the PRF and some for actually computing the secret K_{AB} *using* protected secrets. For computing pairwise secrets facilitated by probabilistic KPSs like SI schemes or SKIT, the HSMs will only need to be equipped with a hash function (or a block-cipher), registers for storing some protected secrets, and simple control circuitry to reuse the block-cipher/hash for different types of computations (bulk encryption, PRF, random number generation, etc.), and some I/O registers to facilitate exchanges with the device A_D.

1) Facets of Complexity: We shall represent by $\mathcal{S}$, the storage complexity borne by the external resource A_S.

In Model I the device A_D performs PRF computations of complexity $\mathcal{C}_U$ and fetches some encrypted secrets from A_S (flash card). Let $\mathcal{F}$ be the number of bytes fetched from the flash card. The device A_D then provides $\mathcal{F}'$ bytes to the HSM A_M.

In Model II A_S performs PRF computations of complexity $\mathcal{C}_U$ and sends $\mathcal{F}'$ bytes to A_D. These $\mathcal{F}'$ bytes are simply relayed by the device A_D to the HSM A_M.

In both models the HSM complexity remains the same. In both models the HSM receives $\mathcal{F}'$ bytes as inputs (written into the input register of the HSM by the device A_D). For verification of PRFs the HSMs will need to re-evaluate some PRFs. We shall denote this complexity as $\mathcal{C}_M^U$. The HSMs will also need to perform some block-cipher / hash operations. Let $\mathcal{C}_M^R$ represent the number of such operation required to compute a specific pairwise secret. We shall also represent by $\mathcal{S}_M$ the number of secrets that need to be stored inside the HSM.

C. KPS Overheads

Any key distribution scheme for scalable networks requires controlling bodies like i) registration authorities (RA) who verify/assign credentials and/or unique identities to nodes, and ii) KDCs which *induct* the nodes into the network by providing secrets, and thus enable them to authenticate themselves to other nodes in the network.

From the perspective of the KDCs and RAs, nodes are HSMs housed in different types of devices. RAs verify the integrity of such HSMs, assign a unique identity, and provide a secret to the HSM. This secret will also be conveyed by the RA to the KDC(s) to bootstrap the key distribution process. Let M_A be such a unique secret provided to node A.

Due to the total freedom available in choosing the secrets the KDC can choose a single master secret and compute any of the P KPS secrets on demand, using a secure one-way function. For example, if M_s is the master secret chosen by the KDC, the KDC can generate any of the P SI scheme secrets as $K_{i,j} = h(M_s, i, j)$. For SKIT, the KDC compute any of the mM secrets as $K_{i,j} = h(M_s, i, j)$ and any ticket $T_X(i, j)$ as $h(h(M_s, i, j), X)$.

For inducting a node with identity A the KDC has to compute the set of $k \approx 8$ million secrets $\mathbb{S}_A$ for SI schemes, and $m = 32$ secrets and $mM \approx 3$ million tickets for SKIT. Let $\mathcal{T}$ denote the complexity of computations performed by the KDC. For both probabilistic KPSs $\mathcal{T} = \mathbb{O}(n \log(1/p))$.

The KDC can provide all k secrets (or secrets and identity tickets for SKIT) by encrypting them using the secret M_A provided to the KDC and the node by the RA. More specifically, for SI schemes the HSM stores one secret M_A (or $\mathcal{S}_M = 1$); a secret $K_{i,j}$ in the key ring $\mathbb{S}_A$ is stored encrypted as $h(M_A, i, j) \oplus K_{i,j}$. For SKIT it is convenient[2] for the HSMs to store $m = 32$ secrets in addition to the secret M_A (or $\mathcal{S}_M = 1+32 = 33$). A ticket $T_A(i, j)$ is stored encrypted as $T_A(i, j) \oplus h(M_A, i, j)$.

The sequence of operations performed by A (or more

[2]For reasons explained later in this paper.

precisely, the HSM A_M, the device A_D and the external resource A_S) to compute a pairwise secret K_{AB}, and the facets of complexity associated with each step, are as follows:

SKIT:

1) $\mathcal{C}_U$: This involves computing $m = 32$ PRFs: $f_M(B, i)$ for $0 \leq i \leq m-1$; this is performed by the device A_D in Model I and the external device A_S in Model II.

2) $\mathcal{F}$: In Model I the device A_D fetches $m = 32$ encrypted tickets from pluggable storage A_S; assuming each ticket is 10 bytes, $\mathcal{F} \approx= 320$ bytes.

3) $\mathcal{F}'$: In Model II, the external device A_S XORs all m encrypted tickets $T_A(i, b_i) \oplus h(M_A, i, b_i)$, $0 \leq i \leq m-1$, and sends a single 10 byte value E_{AB}: or $\mathcal{F}' = 10$ bytes. The device A_D relays E_{AB} to the HSM A_M. In Model I the device A_D XORs all m tickets together and sends the 10 byte value E_{AB} to the HSM.

4) $\mathcal{C}_M^U$: The HSM computes $m = 32$ PRFs $f_M(B, i) = b_i$ (or $\mathcal{C}_M^U$ amounts to $m = 32$ PRF computations).

5) $\mathcal{C}_M^R$: The HSM i) computes $h(M_A, i, b_i)$, $0 \leq i \leq m-1$ and XORs the m values together: let this value be H_B (note that $H_B \oplus E_{AB}$ is simply the XOR of 32 tickets $T_A(i, b_i), 0 \leq i \leq m-1$); ii) computes $T_B(i, a_i) = h(K_{i,a_i}, B)$ for $0 \leq i \leq m$, where the $m = 32$ secrets K_{i,a_i} are stored inside the HSM; iii) XORs the 32 tickets with $H_B \oplus E_{AB}$ to yield K_{AB}. Thus $\mathcal{C}_M^R$ amounts to $2m = 64$ block-cipher/hash evaluations.

SI Scheme:

1) $\mathcal{C}_U$ involves computation of $2k$ (about 16 million) PRFs: computing $f_t(A, i)$ and $f_t(B, i)$ for $0 \leq i \leq k-1$, and determining the subset of k_f indices $\{i_1 \cdots i_{k_f}\} \in \mathbb{Z}_k$ for which $a_i = b_i$. This computation is performed by the device A_D in Model I, and by the active external resource in Model II.

2) $\mathcal{F}$: In Model I the device fetches $k_f = 121$ encrypted secrets from storage. Assuming that each secret is 10 bytes, we have $\mathcal{F} \approx 10k_f = 1210$ bytes.

3) $\mathcal{F}'$: The active external resource in Model II (or the device A_D in Model I) XORs all k_f secrets together (let this 10-byte value be E_{AB}) and sends E_{AB} and k_f indices $\{i_1 \cdots i_{k_f}\} \in \mathbb{Z}_k$ to the HSM. As each index needs $\log_2 k \approx 23$ bits, $\mathcal{F}' \approx 10 + 121 \times 23/8 \approx 358$ bytes.

4) $\mathcal{C}_M^U$: The HSM performs $2k_f = 242$ PRF computations to *verify* that for the k_f indices $\{i_1 \cdots i_{k_f}\} \in \mathbb{Z}_k$ provided by the device, $f_t(A, i) = a_i = f_t(B, i) = b_i$ ($\mathcal{C}_M^U$ amounts to $2k_f = 242$ PRF evaluations);

5) $\mathcal{C}_M^R$: The HSM computes $h(M_A, i, a_i)$ for each common index i and XORs the k_f such values together to yield a 10-byte value H_B. Now $H_B \oplus E_{AB} = K_{AB}$. Thus $\mathcal{C}_M^R$ amounts to $k_f = 121$ block-cipher/hash evaluations.

SKGS:

For SKGS *all* facets of complexity are proportional to the desired collusion resistance n. More specifically, for $n^* = 2^{16}$-secure SKGS: a) $\mathcal{F} = k = n + 1$ encrypted secrets need to be fetched from storage and provided to the HSM (or $\mathcal{F}' = \mathcal{F} = n + 1$; and b) the HSM computational overhead $\mathcal{C}_M$ include i) 1 finite-field exponentiation to compute α^B; ii) $n-2$ finite-field multiplications to compute α^{iB} for $0 \leq i \leq k-1$; iii) $n+1$ decryptions, and iv) $n+1$ finite-field multiplications $\alpha^{iB} \cdot s_A^i$ to compute K_{AB}.

D. Comparison

Table 1 provides a quick comparison of various facets of complexity of KPSs to realize the same (n^*, p^*)-security. In the table the computational overhead for the KDC (for computing the key ring) is $\mathcal{T}$; the external storage complexity $\mathcal{S}$ is indicated in MBs, assuming 80-bit secrets and tickets; $\mathcal{C}_U$ is the number of PRF computations performed by the device (in Model I) or an active external resource (in Model II). The fetch overheads $\mathcal{F}$ and $\mathcal{F}'$ are indicated in bytes.

The value $\mathcal{F}'$ is the number of bytes that need to be provided to the HSM (written into its input register), and thus will influence the HSM complexity. The computational overhead $\mathcal{C}_M$ includes both hash evaluations $\mathcal{C}_M^R$ which employ secrets, and PRF computations $\mathcal{C}_M^U$ which do not employ secrets. For SKGS $\mathcal{C}_M^U$ is number of finite-field multiplications in public computations (for computing α^B and α^{iB} for $0 \leq i \leq k-1$). $\mathcal{C}_M^R$ is the number of finite-field multiplications that need to be performed using KPS secrets (computing $\alpha^{iB} \cdot s_A^i$, for $0 \leq i \leq k-1$). The value $\mathcal{S}_M$, is the number of secrets that need to be stored inside the HSM.

Note that while the complexity facets $\mathcal{T}$, $\mathcal{S}$, $\mathcal{C}_U$ and $\mathcal{F}$ have no bearing on the complexity of the HSM, $\mathcal{F}'$, $\mathcal{C}_M$ (which includes $\mathcal{C}_M^U$ and $\mathcal{C}_M^R$), and $\mathcal{S}_M$ affect the HSM complexity.

For probabilistic schemes resources outside the HSM perform some *public* computations (PRFs) which do *not* require the use of secrets. This is done to enable the external device to choose a small subset of the secrets in the key ring and provide them to the HSM. Secondly, in the case of SI, this can also reduce the complexity of operations performed by the HSM: while the external resource computes 16 million PRFs, the HSM (which does not trust the external device) can still verify the PRF computations by computing a mere $2k_f = 242$ PRFs. While SKGS also includes some public computations (computing α^B and α^{iB} for $0 \leq i \leq k-1$), there is no reason for the external devices to compute them (as all k secrets need to be provided to the device, all of which will be supplied by the device to the HSM).

TABLE I

COMPARISON OF VARIOUS FACETS OF COMPLEXITY OF SKGS, SI, AND SKIT FOR $n^* = 2^{16}$ AND $p^* = 2^{-64}$. THE FIRST FOUR FACETS ($\mathcal{T}, \mathcal{S}, \mathcal{F}$ AND $\mathcal{C}_U$) DO NOT AFFECT THE HSM. THE SECOND FOUR FACETS IMPACT THE HSM.

	SKGS	SKGS-P	SI	SKIT
$\mathcal{T}$	2.1e9	2.1e9	8e6	3e6
$\mathcal{S}$	0.64 MB	–	80 MB	30 MB
$\mathcal{F}$	65537	–	1210	320
$\mathcal{C}_U$	–	–	16,000,000	32
$\mathcal{F}'$	65537	–	358	10
$\mathcal{C}_M^U$	65537	65537	242	32
$\mathcal{C}_M^R$	65537	65537	121	64
$\mathcal{S}_M$	1	65537	1	33

The only practical way to employ SKGS is if all secrets are stored *inside* the HSM (SKGS-P in the table). While this will render the fetch overheads $\mathcal{F}$ and $\mathcal{F}'$ less of an issue, it does nothing to relieve the $\mathbb{O}(n)$ computational overheads $\mathcal{C}_M$ borne by the HSM. For SKGS the HSM computational overhead $\mathcal{C}_M$ becomes comparable to public key schemes even for n of the order of a few hundreds. Thus, in scenarios where we desire large collusion resistance n, there is obviously no reason to even consider SKGS as a possible candidate.

As is clearly evident, every facet of complexity is lower for SKIT compared to SI scheme - except for the modest storage required inside the HSM ($\mathcal{S}_M$ is 33 for SKIT vs 1 for SI). However, if desired, it is also possible to store the $m = 32$ secrets outside the HSM (and thus reduce $\mathcal{S}_M$ to 1). In this case, for computing any pairwise secret, $m = 32$ secrets also need to be fetched (from external storage) and decrypted inside the HSM. Thus, reducing $\mathcal{S}_M$ to 1 will be accompanied by a 320-byte increase in both $\mathcal{F}'$ and $\mathcal{F}$. Furthermore, the HSM computational overhead $\mathcal{C}_M$ will increase from 64 to 96 (due to 32 additional decryptions). Thus, reducing $\mathcal{S}_M$ results in $\mathcal{F} : 320 \rightarrow 640, \mathcal{F}' : 10 \rightarrow 330, \mathcal{C}_M^R : 64 \rightarrow 96$). Clearly, increasing the storage $\mathcal{S}_M$ by a small amount is well worth the reduction in other facets of complexity.

V. DISCUSSIONS AND CONCLUSIONS

Probabilistic KPSs have several advantages over deterministic schemes. Implementation of probabilistic KPSs requires only a block-cipher, which can be used for encryption, hashing and PRF generation. Note that a block cipher (or hash function) is required *in any case* for encryption of messages and computing hashed message authentication codes (HMACs) *using* pairwise secrets. Implementation of finite-field arithmetic required for Blom's SKGS can be more expensive as *additional* circuitry will be required for this purpose. As mentioned earlier, this was one of the main motivations for KPSs based on the idea of subset intersection.

Probabilistic KPSs also have low deployment complexity as they lend themselves readily to i) seamless renewal, and ii) simple strategies for employing multiple independent escrow (KDCs). For SKGS changing even one of the P KDC secrets will result in modification of *every* secret assigned to *every* node. Thus, renewal involves replacing *all* secrets assigned to *every* node. This may be very difficult to achieve without interrupting the regular operation of the deployment. Furthermore, if two n-secure deterministic schemes are deployed in parallel (controlled by two independent KDCs; and each node receives $\mathcal{O}(n)$ secrets from each KDC), the resulting KPS, where shared secrets from both KPSs are used to establish pairwise secrets, is still only n-secure. Thus simple strategies for increasing the number of escrows will result in increase in complexity proportional to the number of escrows.

On the other hand, two (n, p)-secure KPSs can be combined to yield an (n, p^2)-secure KPS. For example, (n, p)-secure SI scheme with parameters (k, t) can actually be s parallel deployments of (n, p_i)-secure schemes with parameters (k_i, t) where $1 \leq i \leq s$, $\prod_{i=1}^{s} p_i = p$, and $\sum_{i=1}^{s} k_i = k$. SKIT with parameters $(s = m, M)$ can actually be s independent deployments under the control of s KDCs (who agree on a PRF). Thus, parallel deployments (controlled by independent KDCs) can be realized *without* any loss of efficiency. To facilitate seamless renewal, $s - 1$ of the s systems can be used during the finite period required for renewing the secrets of one of the s systems.

However, the most compelling advantages of probabilistic KPSs are i) the low fetch complexity $\mathcal{F}$ and $\mathcal{F}'$ and ii) computational overhead for the HSM $\mathcal{C}_M$. More specifically, for probabilistic KPSs $\mathcal{C}_M$, $\mathcal{F}$ and $\mathcal{F}'$ are all *independent* of the desired collusion resistance n. More specifically, for SI $\mathcal{F}$ and $\mathcal{F}'$ are proportional to $\log(1/p)$ (or $k_f = k/t$); For SKIT while $\mathcal{F}$ is proportional to $\log(1/p)$ (the parameter m), $\mathcal{F}'$ is independent of m. Irrespective of the desired n and p the value $\mathcal{F}'$ depends only on the length of each key. The low fetch complexity makes it feasible to off-load storage overhead to external devices. The low computational overhead $\mathcal{C}_M$ makes it possible to employ HSMs of very low complexity, which can be more easily rendered trustworthy.

The primary bottle-neck for SI schemes is the $\mathbb{O}(n \log(1/p))$ complexity $\mathcal{C}_U$ to be borne by resources *outside* the HSM. While external resources can easily afford tens of MBs of storage, millions of PRF computations may be an unreasonable demand. SKIT overcomes this major limitation of SI schemes. Simultaneously, SKIT demands lower storage overhead, lower fetch overhead and lower complexity of computations in the HSM.

A. Related Work

The first probabilistic KPSs was proposed by Leighton and Micali [17] (1994). The first probabilistic KPS based on subset intersections was proposed by Dyer et al [7] (in 1995). Most of the later incarnations of random subset allocation schemes were in the context of sensor network applications [9], [10]. In such schemes, due to random (*not* pseudo-random) allocation of subsets, secrets assigned to nodes are *not* bound to identities. Thus the schemes in [9] and [10] only cater for privacy (and not mutual authentication). Several schemes have also been proposed in the literature with identity-based allocation of subsets of secrets [11], [12]. The primary difference between such schemes lies in the choice of the PRF, which determines *how* the identity is mapped to indices of the keys assigned. While the differences in the choice of PRF affects the public function complexity[3] it does not affect the n/k "efficiency."

Some P-KPSs have also been proposed [13]-[14] which are essentially combinations of Blom's schemes and subset allocation schemes. While inheriting some of the advantages of probabilistic KPSs they also inherit some disadvantages of the underlying deterministic scheme, like the need for finite-field arithmetic, substantially higher $\mathcal{F}$, $\mathcal{F}'$ and $\mathcal{C}_M$ complexity.

B. Conclusions

In application scenarios where there is a need for hardware assisted protection of secrets, computational overhead, especially for operations to be performed inside a trusted boundary is expensive. Reducing the heat dissipated inside the protected boundary can lead to improved strategies for shielding the HSM from intrusions. Lower complexity of circuitry implies better ability to verify the integrity of HSMs. Furthermore, we can then afford more real estate to the components necessary for providing tamper-responsiveness.

While it is well appreciated that KPSs are trade-offs between complexity and collusion resistance, the various facets of complexity, especially when KPSs secrets are protected by HSMs, have not received attention thus far in the literature. Probabilistic KPSs are well suited for devices equipped with HSMs as the collusion resistance of probabilistic KPSs can be increased to any extent without affecting the HSM computational overhead $\mathcal{C}_M$. Synergistically, the low computational overhead improves the ability of HSMs to protect secrets, and further alleviates concerns regarding collusion susceptibility. However, for subset intersection schemes the PRF complexity $\mathcal{C}_U$ increases linearly with the desired collusion resistance n. Even while much of this complexity can be off-loaded to external devices, the PRF complexity remains the bottle neck for achievable collusion resistance n.

We proposed a novel probabilistic KPS, SKIT, which has several compelling advantages over SI schemes. For SKIT even the PRF complexity is independent of n. More specifically, the PRF complexity is reduced by a factor of about $8n$ (for example, if we desire a $n = 2^{17}$-secure scheme the PRF complexity is lower by a factor of *one million*). Only the external storage complexity is proportional to n, and is less than half of that required for SI schemes. The complexity of operations $\mathcal{C}_M^R$ and $\mathcal{C}_M^U$ to be performed inside the HSM are also reduced by a factor 2. The fetch overheads $\mathcal{F}$ and $\mathcal{F}'$ are reduced by factors 4 and 36 respectively. The substantially lower $\mathcal{F}'$ value is particularly advantageous as $\mathcal{F}'$ affects the complexity of the interfaces to the HSM.

[3]The version presented in this paper has the least PRF complexity ($2k$). For other approaches the PRF complexity ranges from $\mathbb{O}(k \log k)$ to $\mathbb{O}(k^2)$.

References

[1] M. Gagne, "Identity-Based Encryption: a Survey," CryptoBytes, Vol. 6, No. 1, pp. 10–19, RSA Laboratories, 2003.

[2] R. Blom, "Non-public Key Distribution," Crypto-82, pp 231–236, 1982.

[3] 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.

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

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

[6] 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.

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

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

[9] L. Eschenauer, V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," Ninth ACM CCS, Washington DC, pp 41-47, Nov 2002.

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

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

[12] 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.

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

[14] 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.

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

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

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