Key Management with Group-Wise Pre-Deployed Keying and Secret Sharing Pre-Deployed Keying

Yee Wei Law, Sandro Etalle, Pieter Hartel
Department of Computer Science, University of Twente
{ywlaw, etalle, pieter}@cs.utwente.nl

July 16, 2002

Abstract

In wireless sensor networks, the key deployment problem has received little attention, whereas it is in fact fundamental, heavily involving crucial (scarce) resources of ad-hoc networks, such as memory and energy availability. In this paper, we first briefly survey the state-of-the art of key deployment strategies that are amenable to ad-hoc network. Then we proposed two possible methods and we shortly investigate their space and computational complexity.

1 Introduction

The objective of the EYES (http://eyes.eu.org) project is to develop self-organizing and collaborative energy-efficient sensor networks. As sensors operate unattended and the communication transmissions occur in an open medium, the system is susceptible to physical attacks and network attacks. Cryptography is the foundational technology for countering these attacks and keys are the centerpiece of cryptography. As part of the project, work is done on the design and implementation of a key management architecture for large-scale (1000+ nodes) sensor networks. *Key management* is the process by which keys are generated, stored, protected, transferred, loaded, used, and destroyed. Keying refers to the process of deriving common secret keys among communicating parties. *Pre-deployed keying* refers to the distribution of key(s) to the nodes before their deployment [2]. *Pairwise keying* involves two parties agreeing on and communicating with a *session key* after deployment, while *group keying* involves more than two parties using a common *group key*. Group keying is important for multicasting. For EYES, we are seeking solutions for *semantic addressing* for mapping terms like "all sensors in the living room" to network identifiers, possibly using multicast techniques for maximum energy efficiency. This is why group communication is important.

In this paper, we investigate various options for key deployment, most notably *group-wise pre-deployed keying* and *secret sharing pre-deployed keying*. Based on our analysis, we propose a group-wise pre-deployed keying scheme called *k-secure t-limited group-wise pre-deployed keying* based on Blundo et al.'s key distribution scheme [1] that is information-theoretically secure. We also propose a secret sharing pre-deployed keying scheme called *private-key sharing pre-deployed keying*. Finally a comparison is made between the two approaches.

2 Key Deployment Design Space

Key deployment is an unsolved problem in wireless sensor networks. At the extremes, there are *network-wide pre-deployed keying* and *node-specific pre-deployed keying* [2]. Network-wide pre-deployed keying equips every node in the network with the same key and equates the compromise of the single system key with the compromise of the entire network. Node-specific pre-deployed keying assigns a unique key to every combination of communicating nodes. In general, to secure groups of up to G nodes in a network of N nodes, $\sum_{g=2}^{G} \binom{N}{g}$ keys have to be generated and deployed so that each node holds $\sum_{g=1}^{G-1} \binom{N-1}{g}$ keys. The security achieved by this scheme is optimal, however the storage requirement is unrealistic.

Between the minimal security of network-wide pre-deployed keying and the maximal security of node-specific pre-deployed keying, lies Blundo et al.'s k-secure t-conference key distribution scheme (KDS) [1]. The goal of this scheme is to guarantee secure communication among any group of up to t nodes against any adversarial coalition of up to k nodes in a n-sized network, where $k \leq n - t$. One of the major merits of this approach is that no actual key is stored, but only the material to construct the keys, i.e. the *keying material.* In the *non-interactive* version of this scheme, all the keying material required for establishing all group keys are pre-deployed, so that every node can evaluate by itself, without interacting with other nodes, the key to communicate with the members of its intended group. The amount of keying material each node has to store is $\binom{k+t-1}{t-1} l_k$ bytes, where l_k is the length of a key in bytes. In the *interactive* version, a node is required to choose a random secret key, apply some transform on the key and communicate the result to the other parties, to establish a common group key. For this case, each node has to store only $(k + t - 1)l_k$ bytes of keying material, but this scheme is only *one-time secure.*

Blundo et al.'s non-interactive KDS achieves a significant reduction in storage space compared with node-specific pre-deployed keying. Nonetheless, it cannot be applied as-is to the EYES system. In fact, if we assume that the percentage of attacker nodes is topped at 10% and all keys are 16 bytes long, for large networks of size $n \geq 1000$ and a maximum group size of $t \leq \frac{9n}{10}$, the amount of keying material required per node is $16\binom{\frac{n}{10}+t-1}{\frac{n}{10}}$ bytes. In other words, the amount of keying material is of the order $O(n^{t-1})$, or equivalently $O(t^{\frac{n}{10}})$, thus t must be kept very small since n is large.

All hope is not lost however. We have looked at ways of deploying the keys such that either (1) there is more than one key throughout the system, or (2) a key is never stored as a whole in any node, but shared across a number of nodes. We call approach (1) *group-wise pre-deployed keying* and approach (2) *secret sharing pre-deployed keying.* These approaches are described in the following sections.

3 Group-Wise Pre-Deployed Keying

The essence of group-wise pre-deployed keying is that the compromise of a group key only compromises the corresponding group and the traffic directed towards that group. By distributing G keys throughout the network, we essentially create G groups of nodes with each group sharing one of the G keys. As the nodes from the same group share the same key, the whole group behaves as if it was one node. By this, we have reduced the task of securing inter-group communications to inter-node communications. Our proposal is to apply the "old trick": Blundo et al.'s k-secure t-conference KDS. We call this scheme *k-secure t-limited group-wise pre-deployed keying*. Now n becomes the total number of groups, t the maximum number of communicating groups, k the number of attacker groups. For a network where $n = 20$, $t = 10$, $k = 2$, a node in any group only needs to store 880 bytes of keying material.

In terms of computational complexity, the computation cost depends on the symmetric polynomial function at the heart of the algorithm. The function has t variables of degree $k + 1$. Fortunately efficient $O(n \log^2 n)$ algorithms for polynomial evaluation exist [7].

The disadvantage of group-wise keying is of course when new groups are added, the existing groups that need to communicate with the new group would have to be installed with new keying material.

4 Secret Sharing Pre-Deployed Keying

Secret sharing pre-deployed keying is essentially a form of (t, n)-threshold secret sharing scheme for the purpose of eliminating the possibility of recovering an entire key wherever it is stored, by dividing the secret key into n shares among which at least $t + 1$ shares are enough to reconstruct the secret key. As the reconstruction of any secret will result in the secret being divulged to the untrusted nodes themselves, secret reconstruction has to be prevented. Our problem thus reduces to how we can make use of the shared key without reconstructing the key itself. If the shared key is a symmetric key, there is no known way of encrypting or decrypting a message with its shares. Even if there is, it will be terribly inefficient to have $t + 1$ nodes collaborating in every encryption or decryption. If the shared key is the private key of a public-private key pair (e.g. RSA), $t + 1$ nodes with different shares can collectively sign (and decrypt) messages provided that the signature function satisfies this property [3]: there exist functions η' and g' such that

$$g_{input}(key) = \eta'(g'_{input}(share_1), \ldots, g'_{input}(share_{t+1})) \quad (1)$$

The RSA signature function is homomorphic, and thus satisfies this property. The bottomline of this insight is that a distributed public-key infrastructure that is similar to Luo et al.'s *local trust model* [5] can now be established. In this model, every node has a public-private key pair and is only trusted if it owns a certificate (carrying its identity and public key) signed by $t + 1$ trusted neighbours. For pairwise keying, Diffie-Hellman-derived or RSA protocols can be used, whereas one of the many multi-party Diffie-Hellman protocols can be used for group keying. As the storage requirement of the key shares and the computational complexity of the shares generation and signature generation algorithm depend on the particular threshold secret sharing scheme employed, we will discuss in the section below some of the options available.

5 Threshold Secret Sharing Schemes

There are currently various threshold cryptographic schemes, from which we have yet to find an optimal solution. Due to space limitation, we only discuss two below:

(1) Shamir's secret sharing scheme uses $t+1$-degree polynomials to generate shares [7]. Joint signature is accomplished through Lagrange interpolation, exponentiation and a technique called *K-bounded coalition offsetting* [4]. The size of each share is the same as the length of the shared private key which is typically 128 bytes long. Computation-wise, key generation only involves polynomial arithmetic, and thus can be performed using efficient $O(n \log^2 n)$ algorithms [7]. The cost of joint signature generation is the sum of $O(1)$ exponentiation, $O(t + 1)$ modular multiplications and $O(t + 1)$ RSA verifications.

(2) Rabin's threshold RSA signature scheme generates shares using RSA key generation coupled with a Verifiable Secret Sharing (VSS) protocol [6]. The joint signature is computed as $SIG(m) = m^{d_{public}} \prod_{i=1}^{t+1} m^{d_i} \pmod N$, where d_i is the individual share and d_{public} is the public share. The size of each share is, up to a small constant, the size of the RSA modulus which is typically 128 bytes. Computation-wise, the cost of key generation is the sum of $O(1)$ RSA secret key generation and $O(n)$ Verifiable Secret Sharing (VSS) protocol executions. The cost of joint signature generation is the sum of $O(t + 2)$ exponentiations and $O(t + 1)$ modular multiplications.

6 Conclusion and Future Work

We have investigated the options available for pre-deployed keying, and arrived at two proposals: *k-secure t-limited group-wise pre-deployed keying* and *private-key shares pre-deployed keying*. Currently, we have estimates of the storage requirements of the keying material, and the computational complexity of the algorithms. However we have yet to determine the computational resource (code size, data size), and energy requirements of our proposals. In conclusion, we have put forth two proposals that seek to find a comfortable trade-off between low resource requirements and high level of security.

References

[1] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 146(1):1–23, 1995.

[2] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical Report #00-010, NAI Labs, 2000.

[3] Y. Desmedt. Some recent research aspects of threshold cryptography. In E. Okamoto, G. Davida, and M. Mambo, editors, *Proceedings on Information Security*, volume 1396 of *LNCS*, pages 158–173. Springer-Verlag, 1997.

[4] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *IEEE International Conference on Network Protocols*, 2001.

[5] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *The 7th IEEE Symposium on Computers and Communications*, 2002.

[6] T. Rabin. A simplified approach to threshold and proactive RSA. In Krawczyk, editor, *CRYPTO '98*, volume 1462 of *LNCS*, pages 89–104. Springer-Verlag, 1998.

[7] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.