Ad Hoc Key Management Infrastructure

B. Lehane[1] L. Dolye[1] D.O'Mahony[2]
[1] Dept. E.&E. Eng., [2] Dept. C.S.,
Trinity College Dublin.
lehaneb@tcd.ie

Abstract

Key Management Infrastructure in an ad hoc network needs to be formed in an ad hoc manner. This paper introduces a novel multipurpose ad hoc Key Management Infrastructure (KMI) that can be used to provide Symmetric Key Infrastructure (SKI) or Public Key Infrastructure (PKI). A further application to group keying with distributed group membership management is also discussed.

***Index Terms*—Ad hoc networks, Security, Key Management, PKI, SKI.**

1. Introduction

In this paper a novel key management infrastructure (KMI) framework is presented. This *ad hoc* KMI framework facilitates the creation of a PKI or SKI in an *ad hoc* manner, i.e. without the requirement of pre-existing infrastructure. An SKI is a novel solution for key management in an ad hoc networking environment. An important application of the SKI is a practical method of group keying that incorporates *distributed* group membership management, something that no previous group keying scheme has addressed. This paper extends the work in [10] which presents a PKI based on a threshold Certificate Authority that can be created in *ad hoc* manner.

Section 2 discusses a *threshold KDC* as the basis for a Symmetric Key Infrastructure. Section 3 briefly describes the notion of a *distributed pseudo random function* and an instance thereof. Section 4 discusses a more distributed solution, for a KDC service, based on the use of a *distributed pseudo random function*. Section 5 discusses the creation of the KDC service and hence the SKI in an *ad hoc* manner. Section 6 discusses the application of this SKI to create a group keying scheme with integrated *distributed* group membership management. Section 7 presents timing results for the implementation and operation of the generic *ad hoc* KMI. The paper concludes in section 8.

2. Threshold KDC

No previous work suggests a solution for a Symmetric Key Infrastructure (SKI) in an ad hoc network. Clearly a centralised Key Distribution Centre (KDC) is not a suitable means of sustaining an SKI in an ad hoc network. An alternative such as Gong's [7] *threshold KDC* is a candidate solution. Gong aims to distribute the role of the KDC to increase availability but uses a *threshold* scheme to maintain an adequate level of security. Gong's scheme uses threshold secret sharing to hide a key exchanged between two end-parties from the servers (which may be attacked/corrupt) used in the transaction. Figure 1. describes Gong's [7] scheme.

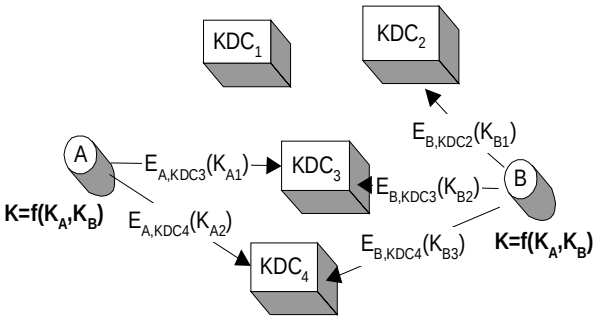

Figure 1. Gong's Threshold KDC

Gong's *threshold KDC* scheme works as follows. When Alice wants to establish a key with Bob, she creates her contribution K_A to a session key. This contribution is divided into n shares using threshold secret sharing [18]. Bob does the same. Alice's shares are labeled K_{Ai} (Bob's are labeled K_{Bi}). Then Alice sends n different shares to Bob via n different KDC servers (and Bob does the same). $E_{A,KDCj}(K_{Ai})$ represents the encryption of Alice's i-th share K_{Ai} with

the symmetric key shared by Alice and KDC_j. The KDC server KDC_j decrypts the key share from Alice, encrypts it with the key he shares with Bob, to give $E_{B,KDCj}(K_{Ai})$, and transmits it to Bob. When both Alice and Bob have collected a threshold *t* of shares they can reconstruct each others key contribution and derive a common session key as a function of these to keys i.e., $K=f(K_A,K_B)$. The KDC acts as an authentication service for the key establishment protocol between the two parties.

Security and availability of this *threshold KDC* service are improved with respect to a single centralised KDC solution. The threshold scheme prevents an attacker from learning keys without compromising at least a threshold *t* of KDC servers. Availability is improved by the increased number of servers providing the service. However, as each KDC server must maintain a keying relationship with every node in the network, this creates an unnecessarily large burden on the KDC service. Also, note that although the service is distributed, all key exchanges involve communications between Alice and Bob and a threshold *t* of the *same* servers. Therefore even if the number of servers is very large, i.e. *n* is large, both parties still have to be able to avail of the *same* threshold *t* of servers to complete the key exchange. Thus the scheme is not as distributed as it may superficially appear.

3. Distributed Pseudo Random Function

Naor [16] introduces the notion of a *distributed pseudo random function.*

A *distributed pseudo random function* is a function *f* which is an approximation to a random function, such that only authorised subsets of players are able to compute the function.

A user who wants to compute *f(x)* sends *x* to the players in an authorised subset and receives information which enables him to compute *f(x).* The type of *authorised subset* is known as the *access structure*. An example of an *access structure* is a *threshold scheme*.

Naor [16] presents various instances of *distributed pseudo random functions* and desirable properties for efficiency of these functions. Desirable properties include *single round of communications, no communication between servers, obliviousness of other servers computing function, robustness.* These four properties describe the properties of a non-interactive threshold decryption/signature algorithm as described in Shoup's paper [19]. Note that, during signing of a public value *x* with a threshold RSA signature protocol a *distributed pseudo random function* is being evaluated by the participants in the threshold RSA signature protocol. Daza [4] proves that an threshold signature protocol that is *existentially secure* against *adaptive chosen message attack* can provide *distributed pseudo random function* that is *semantically secure* in the *random oracle model*. See [4] for details. The security of the *distributed pseudo random function* evaluation is inherited from the underlying threshold RSA signature protocol.

4. Distributed KDC Service

By realising that the threshold RSA signature on a public value is an evaluation of a *distributed pseudo random function* it can be seen that Key Distribution Center can be provided in simple manner. The signing/decryption of a public value, *x*, is the output of a *distributed pseudo random function f(x)* as defined in Naor's [16] paper. The symmetric key distributed by the KDC can simply be a decryption/signature (or the requisite number of bits thereof) on a *public value* that is known by the two communicating end-parties. For security purposes a cryptographic hash of the signature is used, see [4].

Gong's scheme in section 2 is called a *threshold KDC* in this paper to distinguish it from Naor's *distributed KDC*. Naor's scheme will be described further here in terms of the threshold RSA signature protocol used to provide the *distributed pseudo random function* that underpins the KDC service proposed in this paper.

The KDC service consists of a number of servers *n*. Each server holds a share of an RSA private key that will be used in a threshold RSA signature protocol. When two parties (or multiple parties in the case of the group keying scheme described in section 6) wish to establish a symmetric key between them, they query the KDC service. To query the KDC service the two parties contact any *t* servers (not necessarily the same ones) and ask for a result, a signature/decryption, on a common input. Because of the consistency property of the *distributed pseudo random function* the output of *t* servers anywhere in the system will be the same. Thus, the *t* servers used by each of the two parties needn't be

the same. This provides a highly distributed solution for a KDC service.

Each party can contact any *t* servers in his locality. There is no need for servers to interact to maintain consistency on their responses. Servers do not need to maintain consistent information between each other using expensive replication protocols [17], in order for the output of the service to be the same from any *t* servers. The deterministic nature of the threshold RSA protocol ensures that both clients receive the same output.

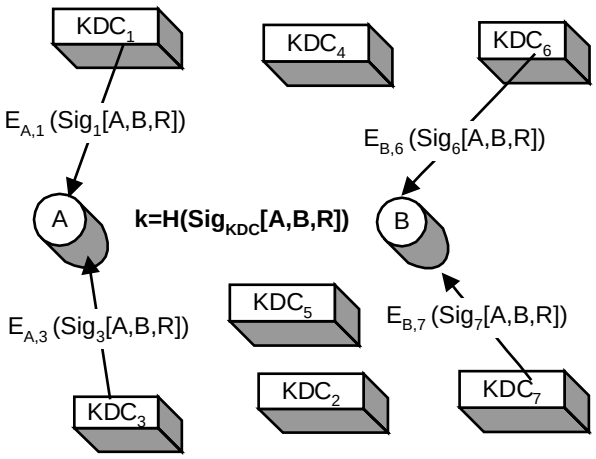

Figure 2. Distributed KDC

This highly distributed ad hoc KDC is depicted in Figure 2. Again a (2,3) threshold scheme is used as an example. Two nodes A and B establish a common secret key via the KDC service. They query two members from this service for a partial signature on a public value [A,B,R] which denotes a concatenation of the identity of the two players and some random data to distinguish one key establishment session from an other. The notation used is as follows $E_{A,1}$ $(Sig_1[A,B,R])$ represents the response from a KDC server KDC_1 to A. This contains the partial signature, denoted Sig_1, from KDC_1 encrypted using the key shared between the KDC server one and the client A. The shared symmetric key derived $k=H(Sig_{KDC}[A,B,R])$ is a cryptographic hash, denoted H, of the *distributed KDC* service's signature on [A,B,R].

5. Ad Hoc SKI

A *complete Threshold RSA protocol* is defined here as a combination of a shared RSA key generation protocol [1] and a compatible threshold RSA signature protocol. It is noted that by using a complete threshold RSA scheme this highly distributed solution for a symmetric key distribution center can be formed in an *ad hoc* manner.

In [10] the implementation of a *complete Threshold RSA scheme* is discussed which combines the shared RSA key generation protocol of Boneh [1] with a non-interactive threshold signature scheme based on Shoup's scheme [19]. The goal in [10] is to create a Public Key Infrastructure in an *ad hoc* manner. That is, the formation of the Public Key Infrastructure mirrors the construction of the underlying network infrastructure, it is formed in an *ad hoc* manner without the assumption of fixed prior infrastructure, such as a trusted third party dealer that generates and distributes the RSA key sharing. Similarly the Symmetric Key Infrastructure proposed here can be formed in an *ad hoc* manner. This same *complete Threshold RSA scheme* can be used as an *ad hoc distributed key distribution centre*. This *ad hoc distributed key management centre* is a novel contribution to the work on key management in ad hoc networks. The SKI can be created without the need for pre-existing security infrastructure and can be operated in a highly distributed manner.

Implementation Problems: What will the *pre-image* of the key be? How will the two clients of the KDC service agree a common public value? One possibility is for the public value to be a concatenation of their identities and a monotonically increasing session number to maintain the freshness of the communications. Further implementation discussion is left for future work.

It is noted here that a further novel use of threshold RSA may be the combination of a *distributed KDC* service and a *threshold Certificate Authority* service into one hybrid symmetric/asymmetric Key Management Infrastructure service. That is, the servers holding the threshold sharing of the RSA private key may offer both services with the same RSA private key. The servers can offer both symmetric key establishment functionality or public key certificate functionality simultaneously. This novel application is called a *hybrid symmetric/asymmetric key management infrastructure* and may be of independent interest outside the field of ad hoc networks.

6. Group Membership Management

In this section a novel group keying scheme is proposed. This scheme makes use of the *ad hoc SKI* designed in this paper. It tackles a problem that is not addressed in the literature, namely, the effective management of group membership and keying in an ad hoc network. The novel solution presented here provides *distributed group membership management* integrated with the *group keying* scheme.

Group keying schemes in the literature that directly refer to ad hoc networks [1], [12] favor collaborative multiparty versions of Diffie-Hellman key exchange [6]. Such a scheme avoids a single point of failure problem when using a group leader, acting as a centralized KDC, to authenticate and key group members.

Using a contributory key agreement protocol to establish a group key removes the obvious choice or method of managing group membership. Typically, collaborative group key agreement papers treat membership management as a problem that is "orthogonal to key establishment and largely a matter of policy" [2]. Others [12] assume *collaborative group membership agreement* protocols are used. In either case establishing who the valid members of the group are, i.e. the current *group view*, is a non-trivial task in an ad hoc network.

6.1. Distributed Group Membership Management

The group keying scheme described in this paper provides a group keying scheme that facilitates *distributed group membership management.* There is no necessity for a group leader to maintain a *group view* or for expensive *collaborative agreement protocols* between all group members to establish a common *group view.*

In this scheme, a *local partial membership decision* reflects an individual player's *partial group view.* This *partial group view* is built by the player based on any local information the player holds himself. An example of local information held by the player himself could be feedback from an Intrusion Detection System (IDS) [14]. A threshold *t* of favorable *local partial decisions* constitutes a valid group member and when combined reveals the group key to that valid member. Therefore, a threshold of *local partial membership decisions* gives a system wide *membership decision.*

The scheme is devised as follows. During group formation a group of nodes (a subset or the entire group) engage in a shared RSA key generation procedure. They generate a threshold sharing of an RSA key pair. This shared RSA key is used to provide the *distributed group membership management and keying* function.

A player P_i obtains the group session key G_k used for group communications by combining a threshold *t* of partial RSA signatures/decryptions on a public value. The *public value* chosen may, for example, be the group name and time or session key number. Let *h(PublicValue)* represent the *pre-image* of the group key. The output of the threshold RSA signature protocol on this value is the group key. This can be computed anywhere in the system by any *t* players in the system.

Each partial signature obtained by the player P_i is a favorable *local partial membership decisions* obtained from another player with a *local group view* which contains this player P_i. Group membership management is therefore distributed throughout the group, rather than being provided by a group leader/chair, nor does it involve distributed consensus between all members of the group. Agreement or consensus is a non-interactive process arrived at as the result of a threshold of favorable decisions from a threshold of group members. This provides a completely distributed membership management scheme that is integrated with group key delivery. Because of the potentially highly dynamic nature of groups in ad hoc networks, it is preferable to have such a distributed scheme that allows a rapid response time to group changes.

How are membership joins and leaves handled? When a member leaves a group, either voluntarily or through expulsion, the group session key must be updated to exclude that member from the group. The nodes in the immediate vicinity are best positioned to decide when a node should be expelled from the group. They can refuse to decrypt further group keys on behalf of the node in question. This provides an immediate localized response to group expulsion. If the RSA group membership management key is shared amongst all members of the group rather than a privileged subset then proactive share refreshing [9] will be required to invalidate this player's share of the

RSA private key. Further discussion of the practical issues of implementing this scheme is left for future work.

The group key distribution scheme described above is a special instance of the more general notion of a *distributed key distribution centre* described in section 4. The group keying scheme uses group members to form the *distributed key distribution centre*. Clients of the service are also partial providers of the service. This is akin to Luo's [11] highly distributed threshold CA where the members of the network form part of the TCA service.

7. Implementation Results

Timing results, which demonstrate the feasibility of forming and operating an ad hoc *KMI*, are presented in this section.

A prototype *ad hoc KMI* was implemented on three ad hoc nodes in a test-bed ad hoc network [13]. The three nodes in the ad hoc *KMI* ran on IPAQ PDAs The model number for the IPAQs are Compaq iPAQ 3950. These devices have a StrongArm 400Mhz processor and 64Mb of memory. The 802.11b network cards used were Lynxsys 802.11b running in ad hoc mode at 11Mbps.

Figure 3 shows the timing results of 200 runs of the shared RSA key generation protocol for both a 1024bit RSA modulus and a 512bit RSA modulus. These results represent the time taken for the ad hoc creation of the KMI. The timing given in Figure 3 is for the generation of the public modulus N. As can be seen, there is no deterministic time in which the protocol runs. It depends on the number of iterations that must be run before a suitable RSA modulus is found. Figure 3 shows that the minimum time measured for a run of the protocol from 200 runs is 5.64 seconds (1024bit key), and 2.03 seconds (512bit key).

Runtime(in secs)	512 bit	1024 bit
Average	98.10	293.13
Minimum	2.03	5.64
Maximum	587.63	1575.75

Figure 3. Shared RSA Key Generation Timing

However the maximum run time is approximately 26 minutes (1575.75) and 10 minutes (587 seconds), respectively. The average run time is the best indication of how long the protocol takes to run. This is 293.13 seconds (O(4.8 minutes)) and 98.1 seconds (O(1.6 minutes)), for the 1024bit and 512bit keys, respectively. The results for the time taken for shared RSA key generation shows that on average in the above described scenario it will take under 2 minutes (512bit key) or under 5 minutes (1024bit key), to generate the RSA key pair to be used by the distributed ad hoc KMI service.

These results are in keeping with Malkin's single threaded implementation [15] of an additive (n,n) shared RSA key generation protocol. This is to be expected as it is the generation of N that is the predominant part of the protocol and the same protocols for this stage have been used here and in Malkin's implementation. Malkin shows in the specific case of a 1024bit key that running time can be decreased by a factor of 4 by adding 6 threads to the protocol. A similar increase in running time of the protocol would apply to the protocol implemented here. For the purpose of the following discussion the running times achieved in our implementation will be used rather than possible/predicted running times.

Assuming a life-time of an ad hoc network to be, say, two weeks or even 24 hours, the set-up of a KMI for the duration of the network taking around 5 minutes (in the 1024bit case) appears to be a reasonable time frame. It is felt that these results on modest computing equipment (3*IPAQ) nodes show that such a key management solution is viable. Certainly these results show that it is at the very least possible to use such a solution in scenarios where such security is demanded or required.

Shared RSA key generation timing results show that the ad hoc inception of a KMI service based on threshold RSA is feasible. Running the KMI service has not been discussed. Timing results for individual partial signature generation operations takes of the order of O(250)ms for 1024bit keys O(50)ms for 512bit keys. Thus computational overhead for the creation of responses from the KMI service is small. Combining these signatures into a single result, is done by the client, this takes O(50) ms. These results demonstrate that a KMI service can potentially service a large number of nodes in real-time.

The timing results presented here for shared key generation and threshold signature generation are based on a non-robust complete threshold RSA scheme as described in [10]. This scheme is secure in the *honest-but-curious* [1] (or passive) adversarial

model. Both Damgard [5] and Foque [8] provide detailed security proofs for a similar *complete Threshold RSA* scheme that is secure in the *active* adversarial model, however no implementation in this model exists. Further work on an implementation and timing results for the far slower robust scheme is required.

8. Conclusions

This paper has introduced the novel notion of an ad hoc Symmetric Key Infrastructure (SKI). It has been shown that the same techniques that are used to form an ad hoc Public Key Infrastructure can be used to form an ad hoc SKI. Thus a *complete* threshold RSA scheme can serve to provide a multipurpose *ad hoc* KMI. This provides the potential for a novel hybrid scheme and a number of variations have been suggested. A further contribution to the literature has been the design of a novel group keying scheme for ad hoc networks based on this ad hoc SKI that integrates *distributed membership management* with the group keying scheme. Timing results for the creation and operation of a prototype ad hoc KMI have been presented to assess and demonstrate the feasibility of such a service.

Acknowledgements

This material is based upon work supported, in part, by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory, under Contract No. F61775-01-WE052.

References

[1] N. Asokan, P. Ginzboorg, 2000, 'Key Agreement in Ad-Hoc networks', Computer Communications, vol. 23, pp. 1627-1637.

[2] G. Ateniese, M. Steiner, G. Tsudik, 1998, 'Authenticated Group Key Agreement and Friends', *Proceedings of the 5th ACM Conference on Computer and Communication Security*, Nov. 02-05, 1998, ACM, New York, pp. 17-26.

[3] D. Boneh, M. Franklin, 1997, 'Efficient Generation of Shared RSA Keys', *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology – Crypto '87*, 1987, Springer-Verlag, LNCS 1294, pp. 425-439.

[4] V. Daza, J. Herranz, G. Sáez, 2003, 'Some Protocols Useful on the Internet for Threshold Signature Schemes', *Proceedings of the 14th Annual International Workshop on Database and Expert Systems Applications (DEXA'03)*, Sep. 1-5, 2003, pp. 359-364.

[5] I. Damgard, M. Koprowski, 2001, 'Practical Threshold RSA Signatures Without a Trusted Dealer', *Proceedings of the Advances in Cryptology – Eurocrypt '01: International Conference on the Theory and Application of Cryptographic Techniques*, May 2001, Springer-Verlag, LNCS 2045, pp. 152-165.

[6] W. Diffie and M. E. Hellman 1976, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644.

[7] L. Gong, 1993, 'Increasing Availability and Security of an Authentication Service', *IEEE Journal on Selected Areas in Communications*, Vol.11, No. 5, pp. 657-662.

[8] P. Foque, J. Stern, 2001, 'Fully Distributed Threshold RSA under Standard Assumptions', *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology – Crypto '01*, Dec. 09-13, 2001, Springer-Verlag, LNCS 2248, pp. 310-330.

[9] A. Herzberg, S. Jarecki, H. Krawczyk, M.Yung, 1995, 'Proactive Secret Sharing or: How to Cope with Perpetual Leakage', *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology – Crypto '95*, 1995, Springer-Verlag, LNCS 963, pp. 457-469.

[10] B. Lehane , L. Doyle, D. O'Mahony, 'Shared RSA Key Generation In A Mobile Adhoc Network', *In proceedings of IEEE 2003 Military Communications Conference (MILCOM2003)*, Boston,MA,USA,Oct 2003.

[11] H. Luo, S. Lu, 2000, 'Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks', *Technical report UCLA-CSD-TR-200030*, October 2000, UCLA.

[12] Y. Kim, A. Perrig, G. Tsudik, 2000, 'Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups', In: S. Jajodin, P. Samarati, (eds), *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Nov. 1-4, 2000, ACM, New York, pp. 235-241

[13] D. O'Mahony, L. Doyle, 2001, 'An Adaptable Node Architecture for Future Wireless Networks, in Mobile Computing: Implementing pervasive information and communication technologies', Kluwer series in Interfaces in OR/CS, Kluwer Publishing, August 2001.

[14] S. Marti, T. Giuli, K. Lai, M. Baker, 2000, 'Mitigating Routing Misbehaviour in Mobile Ad-Hoc Networks', *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000),* Aug. 6-11, 2000.

[15] M. Malkin, T. Wu, D. Boneh, 1999, 'Experimenting with Shared Generation of RSA Key', Proceedings of the Internet Societies' 1999 Symposium on Network and Distributed System Security (SNDSS), pp. 43-56.

[16] M. Naor, B. Pinkas and O. Reingold, 1999, 'Distributed Pseudo-Random Functions and KDCs', Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592, Springer-Verlag, pp. 327-346.

[17] F. Schneider, 1990, 'Implementing Fault-Tolerant Services using the State Machine Approach: A Tutorial', *ACM Computing Surveys*, Vol. 22, No. 4, pp. 299-319.

[18] A. Shamir, 1979, 'How to Share a Secret', Communications of the ACM, Vol. 22, pp. 612-613.

[19] V. Shoup, 2000, 'Practical Threshold Signatures', Proceedings of the Advances in Cryptology – Eurocrypt 2000: International Conference on the Theory and Application of Cryptographic Techniques, May 2000, Springer-Verlag, LNCS 1807, pp. 207-220.