A Distributed Public Key Creation and Distributiuon System For Ad-Hoc Groups

Brian Padalino
(bcp5143@cs.rit.edu)

January 12, 2006

Prof. Hans-Peter Bischof (Chairperson)

Prof. Stanisław Radziszowski (Reader)

Abstract

Group security is a relatively difficult problem especially when dealing with ad-hoc networks. More specifically, managing secure group communications in an ad-hoc situation becomes an extremely difficult problem to try to deal with. There have been some recent publications regarding the establishment of threshold RSA schemes in group situations. I propose implementing the distributed RSA creation method as described by Boneh along with a distribution of the created keys for group management. Moreover, the M2MI middleware will be targeted since it applies directly to ad-hoc networks, distributed systems and broadcast group communication.

Background

The field of ad-hoc networking has grown significantly within the last few years. With this new type of networking comes new challenges in using conventional networking concepts in an ad-hoc formation.

One of the largest challenges appears to be secure data communication between a newly formed ad-hoc group of semi-trusted nodes. The problem lies in the fact that, unlike conventional networks, no public key infrastructure (PKI) exists. Conventionally, the PKI is used to authenticate, sign certificates and pass session keys back and forth between client over public channels in a secure manner. Being able to securely pass session keys to a secured group in an ad-hoc formation is something that ends up being both desirable and relatively complicated.

The proposed solution here uses the homomorphic property of the RSA algorithm. After an ad-hoc group of semi-trusted nodes are formed, and they decide they wish to communicate securely, the Boneh Distributed RSA algorithm can be used to create a group that has an encryption exponent, allowing any member of the group to encode a message but require some cooperation between the members to decrypt the message. In this case, the message will always be a session key coming from the group initiator.

Boneh Group RSA Protocol

The protocol, as described by Dan Boneh, is considered to be secure in an honest-but-curious scenario where the participants involved with the protocol are allowed to choose any value they wish to try to find out more than they are supposed to as long as they still follow the protocol. Moreover, the protocol is $\lfloor \frac{n}{2} \rfloor$ safe requiring that there be more than 2 participants. Other improvements have been suggested to allow generation between 2 parties, but they require highly theoretical oblivious transfer techniques which are used more for theoretical proof as opposed to an implementation suggestion. For the scope of this project, the Boneh RSA protocol is the guideline that will be followed. As described, the protocol consists of 4 major sections:

1. Choosing The Candidates

2. Computing N
3. Biprimality Testing
4. Key Generation

All of the data sharing that occurs follows the classical BGW protocol which uses randomly generated polynomials of specific degrees and solves for their respective coefficients.

Secure Channels

Creating a secure channel using public channels between two users is assumed to be used when communicating securely in a point-to-point fashion. Since the topic at hand deals with creating secure groups in ad-hoc situations, it is assumed that any secure individual communications can be handled at a lower level and is not in the scope of this project.

Choosing The Candidates

Boneh's protocol starts very simply with every participant choosing two random n-bit numbers. These numbers will be the users shares of p and q for the creation of the RSA modulus, N. This procedure step is very straight forward and simple.

Computing RSA Modulus N

To find the product N such that each group members shares, p_i and q_i, are multiplied with no information revealed requires the use of a protocol first described by Benaloh[2]. This protocol is $k-1$ private. Boneh describes the protocol in full detail, but a simple introduction is made here.

1. Each user chooses j random elements, one for each participant, of their private share, y_i, such that the multiplication of the elements yields the private share.
2. Each user sends their private shares, y_i, j to each respective user.
3. Each user receives j elements, multiplies out their shares and sends this information to each of the k users.
4. Each of the end users receives the multiplication of each of the shares and can then compute the multiplicitive share without revealing any information.

This basic protocol is extended to the BGW protocol which, in turn, yields $(\sum p_i)(\sum q_i) = N$.

Biprimality Testing

Testing to ensure N is truly a composite of only two primes is very important and is done in a distributed fashion. For the primality test, it is noted that it only works on Blum integers, or numbers which are 3 mod 4. This can be easily accomplished by having each of the users choose shares which are 0 mod 4 and the secure group initiator choose shares that are 3 mod 4. Boneh's method for biprimality testing utilizes Benaloh's method of multiplicitive sharing as described earlier.

Key Generation

Once the RSA modulus, N, has been computed and passes the testing phase, the encryption exponent, e, can easily be calculated locally and shared.

Boneh describes two methods for calculating shares of d: one which requires e to be small (< 1000), and one which works for all values of e. Needless to say, calculating shares of d is not quite as straight forward as the calculation of e. Full details are explained within Boneh's paper.

Project Description

Since the M2MI framework lends itself very well to group and ad-hoc applications, the project detail is to implement the proposed algorithm within a Java application with the use of the M2MI API for all group communications.

As previously mentioned, secure individual communication is not the primary focus of this project and, therefore, will be assumed to be handled below this application layer. All secure individual communications that are assumed will be noted as per the original protocol.

The application will allow for the creation of multiple groups for broadcast communication, each of which will have the ability to secure their group once it is formed.

The protocol as described by Boneh will be followed for the creation of the group RSA modulus, encryption exponent, and decryption exponents with the aforementioned exceptions in regards to secure individual communications. Not all members of the group will be required to partake in the creation of the group key. During the secure group communication establishment process, any of the members currently active with the group key generation are not allowed to leave the group, though new non-key members can join the group. Once the keys are formed, they may be distributed amongst the other members and cliques for decryption are then formed.

After RSA key formation has occured, and the secure keys are in place, a protocol for delivering session-keys may be implemented. This would be the most desirable encryption mechanism since it mimics real-world applications more than a straight public key encryption scheme does.

Project Scope

This scope of this project will focus on:

- Implementing and understanding the Boneh Distributed RSA algorithm
- Using the M2MI middleware for implementation
- Distributing session keys via newly created RSA system

This scope of this project will *not* focus on:

- Node authentication
- Implementing secure Unihandle M2MI communication

Deliverables

- Application source code
- Javadoc documentation of source classes
- Description of how to build and use the source
- Project report

Metrics

A comparison will be made between a current C implmentation written by Boneh, Malkin, Wu [5]. Specifically, the comparisons will be made between groups of 3 and 4 without using SSL while trying to generate a 512-bit modulus. The values are presented below and are measured in milliseconds unless otherwise noted.

Due to the fact that multicast routing is not available throughout the Internet reliably, all tests will be performed on a local area network and not over the public Internet.

Clients	Sieve time	BGW time	Trial div. time	Prime Test Time	Iteration time	Total time
3	326	413	2.2	628	695	1.51 min
4	689	861	1.5	2035	1707	3.70 min

TABLE 1: Distributed Metrics

Given the current measurements are taken with respect to time and raw computing power, these exact measurement cannot be compared directly with the project. Instead, ratios of performance between different portions of the protocol shall be considered.

Timeline/Schedule

The following timeline is a basic outline as to the major milestones along with predicted completion dates.

Date	Milestone
10/1 - 10/31	Model Boneh algorithm sequentially
11/1 - 11/8	Define exported interfaces
11/8 - 12/6	Distribute sequential Boneh model
12/6 - 12/20	Session key creation and distribution
12/20 - 1/10	Perform Testing and metrics
1/10 - 2/10	Write report

TABLE 2: Project Shedule

References

1. Ben-Or, M., Goldwasser, S., and Wigderson, A. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, Ill., May 2-4). ACM, New York, pp. 1-10.
2. Benaloh (Cohen), J. 1987. Secret sharing homomorphisms: keeping shares of a secret secret. *Advances in Cryptology - Crypto '86*. Lecure Notes in Computer Science, vol. 263. Springer-Verlag, New York, pp. 251-260.
3. Boneh, D., Franklin, M. 2001. Efficient generation of shared RSA keys. *Proceedings Crypto '97*. Lecture Notes in Computer Science, vol. 1233. Springer-Verlag, New York, pp. 425-439.
4. Catalano D, Gennaro R, Halevi S. Computing Inverses over A Shared Secret Modulus. *Advances in Cryptology - EUROCRYPT 2000*. Lecture Notes in Computer Science, vol. 1807. Springer-Verlag, New York, pp. 190-206.
5. Malkin M, Wu T, Boneh D. Experimenting with shared generation of RSA keys. *In Internet society's symposium on network and distributed system security (SNDSS)*, 1999; pp. 43-56.
6. Shamir, A. 1979. How to share a secret. *Commun. ACM 22*, 11 (Nov.), 612-613.