1

of/de

1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Methods of Multi-Party Cryptographic Key Establishment

by

Michael K. Just, B.Sc.

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science**

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

August 9, 1994

© Copyright

1994, Michael K. Just

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

THE AUTHOR HAS GRANTED AN
IRREVOCABLE NON-EXCLUSIVE
LICENCE ALLOWING THE NATIONAL
LIBRARY OF CANADA TO
REPRODUCE, LOAN, DISTRIBUTE OR
SELL COPIES OF HIS/HER THESIS BY
ANY MEANS AND IN ANY FORM OR
FORMAT, MAKING THIS THESIS
AVAILABLE TO INTERESTED
PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE
IRREVOCABLE ET NON EXCLUSIVE
PERMETTANT A LA BIBLIOTHEQUE
NATIONALE DU CANADA DE
REPRODUIRE, PRETER, DISTRIBUER
OU VENDRE DES COPIES DE SA
THESE DE QUELQUE MANIERE ET
SOUS QUELQUE FORME QUE CE SOIT
POUR METTRE DES EXEMPLAIRES DE
CETTE THESE A LA DISPOSITION DES
PERSONNE INTERESSEES.

THE AUTHOR RETAINS OWNERSHIP
OF THE COPYRIGHT IN HIS/HER
THESIS. NEITHER THE THESIS NOR
SUBSTANTIAL EXTRACTS FROM IT
MAY BE PRINTED OR OTHERWISE
REPRODUCED WITHOUT HIS/HER
PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE
DU DROIT D'AUTEUR QUI PROTEGE
SA THESE. NI LA THESE NI DES
EXTRAITS SUBSTANTIELS DE CELLE-
CI NE DOIVENT ETRE IMPRIMES OU
AUTREMENT REPRODUITS SANS SON
AUTORISATION.

ISBN 0-315-98564-X

Canada

Name

Mike Just

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer Science

SUBJECT TERM

0984

U·M·I

SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
Ancient 0289
Linguistics 0290
Modern 0291
Literature 0401
General 0294
Classical 0295
Comparative 0297
Medieval 0298
Modern 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
General 0321
Biblical Studies 0319
Clergy 0320
History of 0322
Philosophy of 0469
Theology 0323

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical 0310
Business Administration 0272
General 0770
Accounting 0454
Banking 0338
Management 0385
Marketing 0501
Canadian Studies 0503
Economics 0505
General 0508
Agricultural 0509
Commerce-Business 0510
Finance 0511
History 0358
Labor 0366
Theory 0351
Folklore 0578
Geography 0578
Gerontology 0578
History 0578
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0337
United States 0337
History of Science 0585
Law 0398
Political Science 0615
General 0616
International Law and Relations 0617
Public Administration 0814
Recreation 0452
Social Work 0626
Sociology 0627
General 0938
Criminology and Penology 0631
Demography 0628
Ethnic and Racial Studies 0629
Individual and Family Studies 0630
Industrial and Labor Relations 0700
Public and Social Welfare 0344
Social Structure and Development 0709
Theory and Methods 0999
Transportation 0453
Urban and Regional Planning 0453
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy 0475
Animal Culture and Nutrition 0476
Animal Pathology 0359
Food Science and Technology 0478
Forestry and Wildlife 0479
Plant Culture 0480
Plant Pathology 0817
Plant Physiology 0777
Range Management 0746
Wood Technology 0306
Biology 0287
General 0308
Anatomy 0309
Biostatistics 0379
Botany 0329
Cell 0353
Ecology 0369
Entomology 0793
Genetics 0410
Limnology 0307
Microbiology 0317
Molecular 0416
Neuroscience 0433
Oceanography 0821
Physiology 0778
Radiation 0472
Veterinary Science 0786
Zoology 0760
Biophysics 0425
General 0996
Medical 0996

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
General 0300
Audiology 0992
Chemotherapy 0567
Dentistry 0350
Education 0769
Hospital Management 0758
Human Development 0982
Immunology 0564
Medicine and Surgery 0347
Mental Health 0569
Nursing 0570
Nutrition 0380
Obstetrics and Gynecology 0354
Occupational Health and Therapy 0381
Ophthalmology 0571
Pathology 0419
Pharmacology 0572
Pharmacy 0382
Physical Therapy 0573
Public Health 0574
Radiology 0575
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences 0485
Chemistry 0749
General 0486
Agricultural 0487
Analytical 0488
Biochemistry 0738
Inorganic 0490
Nuclear 0491
Organic 0494
Pharmaceutical 0495
Physical 0754
Polymer 0405
Radiation 0605
Mathematics 0986
Physics 0606
General 0608
Acoustics 0748
Astronomy and Astrophysics 0607
Atmospheric Science 0798
Atomic 0759
Electronics and Electricity 0609
Elementary Particles and High Energy 0610
Fluid and Plasma 0752
Molecular 0611
Nuclear 0756
Optics 0611
Radiation 0463
Solid State 0346
Statistics 0984
Applied Sciences 0346
Applied Mechanics 0984
Computer Science 0984

Engineering 0537
General 0538
Aerospace 0539
Agricultural 0540
Automotive 0541
Biomedical 0542
Chemical 0543
Civil 0544
Electronics and Electrical 0348
Heat and Thermodynamics 0545
Hydraulic 0546
Industrial 0547
Marine 0794
Materials Science 0548
Mechanical 0743
Metallurgy 0551
Mining 0552
Nuclear 0549
Packaging 0765
Petroleum 0554
Sanitary and Municipal 0790
System Science 0428
Geotechnology 0796
Operations Research 0795
Plastics Technology 0994
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

The undersigned hereby recommend to
The Faculty of Graduate Studies and Research
acceptance of the thesis,

Methods of Multi-Party Cryptographic Key Establishment

submitted by

Michael K. Just, B.Sc.

in partial fulfilment of the requirements
for the degree of Master of Computer Science

A handwritten signature in cursive script, appearing to read 'E. Kravtsov', written over a horizontal line.

Director, School of Computer Science

A handwritten signature in cursive script, appearing to read 'E. Kravtsov', written over a horizontal line.

Thesis Supervisor

Carleton University

September 19, 1994

Abstract

The majority of current literature deals with secure communications between two users. This thesis examines a number of protocols that allow groups of more than two users to establish a common key for the purpose of communicating securely. First we examine key agreement protocols whereby users contribute pieces to the construction of a mutual key over insecure channels. Next, we examine secret sharing protocols whereby users are required to physically “pool” together their pieces to establish a key.

Finally, we present methods for broadcasting keys to groups of users. We review methods of serial broadcasting. We then introduce a new model that allows for parallel broadcasting. A lower bound is established for a subset of the parallel broadcast schemes, and we improve upon a scheme such that it achieves this bound. Implementations using secret sharing are reviewed and two new number theoretic schemes are presented.

Acknowledgements

I would like to thank my supervisor, Professor Evangelos Kranakis for his motivation and support. His work ethic and constant questioning of published results provided me with a realistic portrait of what needs to be done to become a good researcher.

I would also like to thank Dr. Paul Van Oorschot for his numerous and lengthy comments and suggestions on the many versions of results that I have submitted to him. He took the time to read everything I gave him, and provided me with extremely helpful and motivational comments.

I also thank Professor Danny Krizanc for his extremely helpful suggestions. His ability to examine any problem, and offer a unique perspective for a solution is truly inspiring.

Contents

Abstract	iii
Acknowledgements	iv
List Of Tables	vii
List Of Figures	viii
1 Introduction	1
1.1 Basic Cryptography	2
1.2 Multi-Party Protocols	7
1.3 Outline	10
2 Multi-Party Key Agreement	11
2.1 Extending Diffie-Hellman	11
2.1.1 Ingemarsson Scheme	13
2.1.2 Burmester-Desmedt Scheme	16
2.1.3 Steer Scheme	19
2.1.4 Comparison of Schemes	20
2.2 Associative One-Way Functions	22
2.2.1 Using AOWFs for Multi-Party Key Agreement	23

3	Secret Sharing	25
3.1	Threshold Schemes	26
3.2	Shamir's Threshold Scheme	29
3.3	Blakley's Threshold Scheme	30
3.4	Reusability of Threshold Schemes	32
4	Broadcasting Session Keys	36
4.1	Serial Broadcasting	38
4.1.1	The Secure Lock	39
4.1.2	Using the Secure Lock	42
4.1.3	The Sealed Lock	44
4.2	Parallel Broadcasting	46
4.3	A Lower Bound	47
4.3.1	Properties of Model	49
4.4	Parallel Broadcasting Using Secret Sharing	53
4.4.1	Using Shamir Interpolation to Broadcast	54
4.4.2	Using Vectors to Broadcast	56
4.4.3	Preventing Insider Attacks	57
4.5	Parallel Broadcasting Using Number Theory	58
4.5.1	Using Quadratic Residues to Broadcast	59
4.5.2	Using Exponentiation to Broadcast	63
4.5.3	Eliminating the Trusted Center	64
4.6	Analysis of Parallel Broadcast Schemes	65
5	Conclusion	67
A	Number Theory	69
A.1	Computing Inverses	69
A.2	Chinese Remainder Theorem	70

References

72

List of Tables

2.1 Ingermarsson’s Multi-Party Key Agreement 15

2.2 Key Agreement Comparison 22

List of Figures

1.1	Conventional Cryptographic System	3
1.2	Public Key Cryptographic System	4
2.1	Ring of 4 users.	14
3.1	Concealer Function	27
3.2	Revealer Function	28
3.3	Blakley's Threshold Scheme	31
4.1	Serial Key Encryption	38
4.2	Parallel Key Encryption	47

Chapter 1

Introduction

The recent surge in computer usage has made possible widespread exchanges of information. The days of physical mail delivery seem to be drawing to an end. Doctors can receive vital patient information at the press of a button. Students and professors can obtain vast amounts of research information without stepping into a library. People can share stock market information from other parts of the world, without ever having physically met.

For many, this information network is taken for granted. The protocols and procedures of making this network possible are left to the various computer programmers. However, one property that should not be overlooked is the security of the communications. For information that is available to everyone, security is usually not a concern. However, some exchanges require the guarantee that only the intended participants have seen the actual information (e.g., patient records, bank transactions).

To achieve this security, one can use cryptography. In this way, participants can use a cryptographic protocol that is shared among them. Most cryptographic protocols allow two users to achieve mutually secure communications. However, many applications require that more than two users share information securely. For example, consider the need for secure cable broadcasting. This involves the

distribution of a secure message from a center to a number of users. Security is also required for telephone communications. Although secure telephones exist, e.g., Motorola STU-III (see [Diff88]), the need for secure conference calls has not been fully examined. The purpose of this thesis is to examine protocols whereby groups of more than two users can achieve secure communications.

1.1 Basic Cryptography

The word cryptography is derived from the Greek words *kryptos*, which means hidden, and *graphein*, which means write. Therefore, cryptography is the study of secret writings.

A *cipher* is a method of transforming *plaintext* or *cleartext* into *ciphertext*. Plaintext is usually a legible form of writing that is recognizable as such because of its redundancy. Ciphertext is an illegible series of text that is apparently unrelated to the corresponding plaintext. *Encipherment* or encryption of a plaintext message produces the corresponding ciphertext. *Decipherment* or decryption of a ciphertext message produces the corresponding plaintext.

The processes of encryption and decryption are performed by the respective functions E and D . Figure 1.1 illustrates how these processes are incorporated into the cryptographic protocol. Using the encryption function E , parameterized by the key K , Alice (A), encrypts the message M to create the ciphertext C , i.e., $C = E_K(M)$. The ciphertext C can be sent over an insecure channel to Bob (B), whereby anyone who has access to the line, can obtain C .

The key K is exchanged beforehand over a secure channel. In this way, B can recover M by inverting the encryption performed by A , i.e., $M = D_K(C)$. The methods of encryption and decryption are usually assumed to be publically known. There are several reasons for this.

1. For users to trust a cryptosystem, they require knowledge of how it works.
2. Many users can use the same system, parameterized by different keys.

Figure 1.1: A conventional or single-key cryptographic system where the enciphering and deciphering operations are performed with the same key.

3. It is expensive to protect an entire cryptosystem, and difficult to create many secure cryptosystems, i.e., one for each pair of users.

Therefore, the security usually depends on maintaining the secrecy of the key K , not on the security of the cryptosystem itself. (Although additional security may be provided by maintaining the secrecy of the cryptosystem, one should not rely on this fact when designing a cryptosystem [Bras88].) The insecure channel (see Figure 1.1) is monitored by an eavesdropper X . X may be *passive* (only listens to the information) or *active* (injects his own information). When X attempts to recover plaintext from ciphertext or solve for the encryption key, we say that X is a *cryptanalyst*.

For the cryptosystem to be secure, it is necessary that the key space is large, and in general each key provides a distinct mapping from plaintext to ciphertext (i.e., different keys should not convert the same plaintext to the same ciphertext). This provides for security from what is known as an *exhaustive search*. In this attack, a cryptanalyst will attempt to recover M by decrypting C , by using all possible keys.

Besides the exhaustive search attack, several other attacks are identified for cryptosystems. They are based on the amount of information available to the cryptanalyst and are listed in order of severity to legitimate users. The last being the most advantageous to a cryptanalyst.

ciphertext only The cryptanalyst has the ciphertext from several distinct encryptions under the same key. From this he is to either compute the key, or any

Figure 1.2: A public or two key cryptographic system.

of the corresponding plaintext.

known plaintext The cryptanalyst has the ciphertext and corresponding plaintext from several distinct encryptions under the same key. From this he is to either compute the key, or decrypt any subsequent ciphertexts.

chosen plaintext The cryptanalyst has the ciphertexts corresponding to several plaintexts of his choosing. From this he is to either compute the key, or decrypt any subsequent ciphertexts.

There are some disadvantages with *conventional* cryptosystems as described above. The first deals with the number of keys held by each user in the cryptosystem. If there are n users, then each pair of users will share a secret key. Of course, a user should not share the same key with two different users since each of these users could decrypt messages that were intended for the other. Therefore, each user shares a distinct key with every other user. In total, this requires approximately n^2 keys to be stored in the entire cryptosystem.

A second problem is the requirement that the keys be exchanged over a secure channel. This problem was solved in 1976 by Diffie and Hellman [Diff76]. The solution is known as *public-key* or *two-key* cryptography. In a public-key cryptosystem

(PKCS), users publish their encryption key, while maintaining the secrecy of the decryption key. This allows any user to encrypt a message for any other user, given only a single key. Only the user with the corresponding decryption key can recover the intended message. This process is illustrated in Figure 1.2.

Some properties of PKCS's are

- Applying the deciphering transformation to the corresponding ciphertext reveals the message, i.e. $D(E(M)) = M$.
- Both E and D are easy to compute.
- Publically revealing E does not provide an easy way to compute D .

Notice how the last point removes the need for a secure channel as the publication of a user's encryption key could equally be accomplished by users distributing their encryption keys over an *insecure channel* to all other users with whom they intend to communicate. A function E satisfying these points is known as a *one-way trapdoor function*. In other words, E is a one-way function to anyone other than the intended receiver of the message, whereby only knowledge of the trapdoor information (i.e., the key parameterizing D) allows inversion of E .

RSA Encryption

In 1978, Rivest, Shamir and Adleman [RSA78] presented the first implementation of a PKCS. The inversion of the encryption function by anyone other than the intended receiver is believed to be based on the difficulty of factoring a product of two large primes. (The ability to factor certain integers would imply a method for solving RSA, yet it is not known whether breaking of RSA would imply a method for factoring.)

To setup his public key, B selects two large primes p and q and computes the product $n = pq$. B then chooses an enciphering exponent e such that $\gcd(e, \phi(n)) = 1$. The inverse d of e is computed using the methods described in Appendix A.1,

i.e., $ed \equiv 1 \pmod{\phi(n)}$, where $\phi(n) = (p-1)(q-1)$. B publishes the pair (e, n) as his public key and keeps d private.¹

Suppose A decides to send an encrypted message M to B , where $0 \leq M \leq (n-1)$ and $\gcd(M, n) = 1$ (the latter constraint will be true for almost all messages M). A first obtains B 's enciphering key (e, n) from the public directory, then computes

$$\begin{aligned} C &= E_{(e,n)}(M) \\ &= M^e \pmod{n} \end{aligned}$$

and sends the ciphertext C to B over an insecure channel. Upon receipt of C , B uses his trapdoor information, d , to compute

$$\begin{aligned} M &= D_{(d,n)}(C) \\ &= C^d \pmod{n}. \end{aligned}$$

The decryption works because of the fact that the product of e and d is equivalent to 1 modulo $\phi(n)$, where the operations in the exponent of M (or C) are done modulo $\phi(n)$, ($ed \equiv 1 \pmod{\phi(n)} \Rightarrow ed = k\phi(n) + 1$). This is illustrated by the following, expanded calculation of the decryption operation:

$$\begin{aligned} C^d &\equiv M^{ed} \pmod{n} \\ &\equiv M^{k\phi(n)+1} \pmod{n} \\ &\equiv MM^{k\phi(n)} \pmod{n} \\ &\equiv M(M^{\phi(n)})^k \pmod{n} \\ &\equiv M \end{aligned}$$

where $M^{\phi(n)} \equiv 1 \pmod{n}$ is valid by Euler's Theorem (see [Denn83]). We will return to a discussion of RSA in Section 1.2.

Key Establishment

In [Diff76], Diffie and Hellman also mention the problem of key establishment. The establishment of a public directory for using a PKCS, solves the problems previously

¹Use of $\lambda(n)$ in place of $\phi(n)$ is the more general requirement, where $\lambda(n) = \text{lcm}(p-1, q-1)$.

mentioned for conventional cryptosystems, i.e., number of keys stored by each user and secure key establishment. It also removes the need for a secure channel for sharing keys (see Figure 1.1). This is referred to as a *key distribution* or *key transfer*.

A second method of establishing a key is through a key agreement. *Key agreement* or *key exchange* is a method that allows users to contribute information in real-time to the generation of a mutually agreed upon conference key. The following key agreement protocol was presented in [Diff76]. Suppose we have two users, identified by A and B . The computations that they will be performing will be done in the multiplicative group Z_p^* with a generator g . User A selects a random $x_A \in Z_p^*$, computes $y_A = g^{x_A} \bmod p$ and sends y_A to B . User B selects a random $x_B \in Z_p^*$, computes $y_B = g^{x_B} \bmod p$ and sends y_B to A .

Given x_A and y_B , A computes

$$\begin{aligned} K_A &= (y_B)^{x_A} \pmod{p} \\ &= (g^{x_B})^{x_A} \pmod{p} \\ &= (g^{x_A})^{x_B} \pmod{p} \end{aligned}$$

while B computes

$$\begin{aligned} K_B &= (y_A)^{x_B} \pmod{p} \\ &= (g^{x_A})^{x_B} \pmod{p}. \end{aligned}$$

In this way, A and B have agreed upon a common key $K = K_A = K_B$. We return to a discussion of key agreement protocols in Chapter 2.

1.2 Multi-Party Protocols

As mentioned earlier, the security of a cryptographic protocol depends upon keeping the key K secret. Therefore, a secure protocol depends upon how securely users establish the key K . Conventional cryptography uses a secure channel to achieve this exchange. Public-Key cryptography uses two keys. The first is stored in a publicly accessible directory and is used to encrypt a message. The corresponding

decryption key is held only by the user for whom the message is intended. Diffie-Hellman key agreement goes one step further by allowing two users, who have never met, nor share or know any keying information about the other, to agree upon a secret key.

The one common point about the methods described above is that they are developed to work for only two people. It can be potentially dangerous to extend protocols for more than two people, given that they are “secure” for just two people. Consider the following examples, using the RSA cryptosystem (see Section 1.1).

Encrypting a message with RSA can be quite expensive, relative to other encryption methods, e.g., DES [DES77]. It might be considered reasonable to select $e = 3$ for the encryption exponent. This allows for quick and inexpensive encryptions. It is also advantageous since all users have the same public encryption exponent. (Given that the users have distinct moduli, they will have distinct decryption exponents.) This removes the need for storing the encryption exponent for each user in a public directory since all users will have the same encrypting exponent e . (However, the distinct moduli of each user require the use of the public directory.) For normal two-party communications, this protocol is “secure”. However, Hastad [Hast85] has shown that this protocol is insecure when sending the same message to more than 2 users.

Suppose users u_2, u_3, u_4 have the respective public encryption exponents $e_2 = e_3 = e_4 = 3$ and moduli n_2, n_3, n_4 . To encrypt a message M for u_2, u_3 and u_4 , user u_1 computes $C_2 = M^3 \bmod n_2$, $C_3 = M^3 \bmod n_3$ and $C_4 = M^3 \bmod n_4$ and distributes these to the respective users.

Given these messages, and the public information, an eavesdropper X can compute $Y = M^3 \bmod n_2 n_3 n_4$ with the Chinese Remainder Theorem (see Appendix A.2). Since $0 < M < n_i$, for $i = 2, 3, 4$, $M^3 < n_2 n_3 n_4$. Therefore, M can be obtained by computing the cube root of Y , using only a ciphertext only attack. (Note that $M^3 \bmod n_3$ does not allow the recovery of M since M^3 will most likely be reduced modulo n , i.e., modular roots are difficult to compute, while normal roots are easy.)

A second way to misuse the RSA cryptosystem for the purposes of multi-user

protocols, would be to use a common modulus (see [Simm83]). Suppose a central authority were to setup an RSA composite n and securely distribute to all users a pair (e_i, d_i) , consisting of a public encrypting exponent and a private decrypting exponent, and makes the composite n public. Suppose now that u_1 is sending the same message to u_2 and u_3 . The encryptions of a message M would be $C_2 = M^{e_2} \bmod n$ and $C_3 = M^{e_3} \bmod n$, each being distributed to the respective users.

However, given C_2, C_3 and the public enciphering information (e_2, e_3, n) , where $\gcd(e_2, e_3) = 1$ an eavesdropper X can determine M . To do this, X first finds s and t such that $e_2s + e_3t = 1$, using the extended Euclidian algorithm (see Appendix A.1). X can then compute the following:

$$\begin{aligned}
 X &= C_2^s C_3^t \pmod{n} \\
 &= M^{e_2s} M^{e_3t} \pmod{n} \\
 &= M^{e_2s + e_3t} \pmod{n} \\
 &= M^1 \pmod{n} \\
 &= M.
 \end{aligned}$$

Therefore, X is able to recover M , once again with a ciphertext only attack. Even worse is that users can recover each others secret information (see [DeLa84]). However, this attack is beyond the context of this example.

The purpose of these two examples is to illustrate the fact that an otherwise “secure” cryptosystem such as RSA, can be abused for the purposes of achieving secure multi-party communications. In other words, what is secure for two users, may not always be secure for more than two users. Therefore, care must be taken when attempting to achieve secure multi-party communications.

In Chapter 4, we will present methods of broadcasting a secure message to multiple users. More specifically, in Section 4.5.2, we use RSA as the method of encryption.

1.3 Outline

In this thesis, we examine how groups of more than two users can securely establish a common key between them. The protocols under which the key establishment takes place are referred to as *multi-party* or *multi-user* protocols. Consider a set U of n users. A subset P of U , known as the privileged set, consists of those users who will be establishing the key, where $|P| = t \leq n$. For simplicity, we let $\{u_1, \dots, u_t\}$ represent the users in P . In some cases, we will assume that each user u_i has shared a secret s_i beforehand with a trusted center T .

The major contributions of this thesis are found in Chapter 4. This work appears in [Just94]. We present ways for a center to broadcast a message to all users such that only a privileged subset of users is able to recover any meaningful information from this message. The meaningful message is the key that the users in P establish for themselves. We examine several previous protocols and subsequently establish a lower bound on the size of the broadcast message for the model of Section 4.3.1. We also introduce a new framework that allows for the generation of parallel broadcast messages, and present two methods of using this framework. The first is based on the ideas from Chapter 3. We improve on a scheme here such that it achieves the aforementioned lower bound. The second method introduces two number theoretic techniques for achieving a parallel broadcast.

In Chapter 2 we begin by examining ways of extending the original Diffie-Hellman key agreement protocol [Diff76] to allow groups of more than two users to establish a key. Thus far, no protocol has been able to extend Diffie-Hellman to allow more than 2 users to establish a key, while maintaining the elegance of the Diffie-Hellman protocol. We also present a model that allows for an elegant protocol and illustrate how to construct the protocol, though no implementation has been found as of yet.

In Chapter 3 we examine the concept of secret sharing. Although normally considered a method of key management, its end result provides the establishment of a key for the participating users. As mentioned, secret sharing can also be altered to satisfy the requirements for the model described in Chapter 4.

Chapter 2

Multi-Party Key Agreement

A key agreement protocol allows a privileged set P of users to interact via broadcast or direct message passes with the common goal of establishing a key that includes an equal contribution from each of the users. The most famous key agreement protocol is that by Diffie and Hellman (see Section 1.1). Each user broadcasts only a single message, based only on a random integer of his choosing and some publicly available information. However, the protocol succeeds only for two users.

As of today, no multi-party key agreement protocol has been able to capture the elegance and simplicity of the basic two-party Diffie-Hellman protocol. In Section 2.1, we examine the attempts made at extending the Diffie-Hellman protocol to more than two users. In Section 2.2, we present a generalization of the key agreement problem that initiates the search for functions satisfying the properties listed in the section. If such functions exist, we show how to construct an elegant key agreement protocol that extends the properties of the original Diffie-Hellman scheme.

2.1 Extending Diffie-Hellman

In the following sections, we examine several attempts at achieving a multi-party key agreement protocol. In each case, the protocol extends the well-known Diffie-Hellman (DH) two-party protocol (see Section 1.1). However, none of these extensions is able to capture the simplicity and elegance of the DH protocol. The problems, and the relative performance of the protocols are examined.

An Ideal Scheme

To create an ideal scheme that captures the elegance of the original would be an impressive achievement. The elegance results from the fact that each user need only broadcast a single message to all other users.

The following protocol was mentioned in [Fiat93]. It achieves the aforementioned elegance but is susceptible to collusions of two or more users (the authors of [Fiat93] were completely aware of this fact). A trusted center T performs the following to initialize the system. T selects primes p and q and the product $n = pq$ is published; primes p and q are discarded. T then chooses an element α of maximal order from Z_n^* maintaining its secrecy. Each user u_i that joins the system is assigned a key $\alpha_i = \alpha^{x_i} \pmod{n}$ for a random x_i , where $\gcd(x_i, x_j) = 1, \forall u_i, u_j \in U$, and α is an element of maximal order in Z_n^* . User u_i keeps α_i secret but will use x_i as their broadcast message.

For a set P of users to establish a conference key K , each user u_i broadcasts their x_i . Once collecting the broadcasted messages from the other users, u_i calculates the key as

$$\begin{aligned} K &= (\alpha_i)^{\prod_{j=1, j \neq i} x_j} \pmod{n} \\ &= \alpha^{x_1 x_2 \cdots x_i} \pmod{n} \end{aligned}$$

This protocol allows each user to broadcast only one message, while allowing each of the users in P to establish a mutually agreed upon conference key K . The drawbacks of this protocol result from the fact that the secrecy of α must be maintained. This requires the presence of a trusted center T for the system. More importantly, as shown in [Fiat93], two users can collude to recover α .

Throughout the remainder of this section, we will be working in the multiplicative group Z_p^* , for some prime p , where p is a prime number and $p - 1$ has at least one large prime factor (see [Pohl78]). (We do this for simplicity, since the operations performed here can be applied to any finite field). We denote by g as one of the many possible generators of Z_p^* . The values α and p are publicly available to all

users. All computations will be performed in Z_p^* . Without loss of generality, we assume that $u_i \in P$, $i = 1, \dots, t$ represent the users in the privileged set P .

2.1.1 Ingemarsson Scheme

The first extension of the basic Diffie-Hellman protocol was presented by Ingemarsson *et al.* [Inge82]. In this system, users communicate over a uni-directional ring network. Each user $u_i \in P$ chooses a random x_i and computes $y_i = g^{x_i} \bmod p$. The protocol for obtaining a conference key K consists of $t - 1$ steps subsequent to the computation of their y_i . Each step involves some computation, and the passing of values to the next consecutive user in the ring. The computations performed by each user are symmetric, i.e. each user performs the same computations at each step.

The symmetric functions $S^{(j)}(I)$ are defined in [Inge82] as follows.

Definition 2.1 *A symmetric function $S^{(j)}(I)$ is the sum of all possible products of j variables x_i with distinct indices belonging to the set I . The largest set I , denoted by Ω , is*

$$\Omega \triangleq \{1, \dots, t\}.$$

Some examples are

$$\begin{aligned} S^{(1)}(\Omega) &= \sum_{i=1}^t x_i, \\ S^{(2)}(1, 2, 3) &= x_1 x_2 + x_1 x_3 + x_2 x_3, \\ S^{(2)}(1, 2) &= x_1 x_2, \end{aligned} \tag{2.1}$$

$$S^{(t)}(\Omega) = \prod_{i=1}^t x_i. \tag{2.2}$$

For a two-party Diffie-Hellman scheme (see Section 1.1) where the two users have the respective secret pieces x_1 and x_2 , the common key is equivalent to the one computed with (2.1), i.e., $K = g^{S^{(2)}(1,2)} \bmod p$. (For the multi-party scheme, we will be using (2.2).)

Figure 2.1: Ring of 4 users.

An important characteristic concerning (2.1) and (2.2) is that j from Definition 2.1 is equal to the cardinality of I . In [Inge82], this is shown to be a necessary condition to prevent a line tapper (with the ability of eavesdropping on multiple communication links) from deriving the key. This is best illustrated with the following simple example.

Suppose we have a system where 4 users communicate along a uni-directional ring (see Figure 2.1) who will compute the key using the symmetric function $S^{(2)}(1, 2, 3, 4)$ (notice that $j < |I|$). As mentioned earlier, for $t = 4$ users, the key computation will require $t = 4$ steps. The following steps are performed by u_2 to allow computation by all users of the key K .

1. Send g^{x_2} for random x_2 to u_3 .
2. Receive g^{x_1} from u_1 . Send $g^{x_1+x_2}$, $g^{x_1x_2}$ to u_3 .
3. Receive $g^{x_4+x_1}$, $g^{x_4x_1}$ from u_1 . Send $g^{x_4+x_1+x_2}$, $g^{x_4x_1+x_4x_2+x_1x_2}$ to u_3 .
4. Receive $g^{x_3+x_4+x_1}$, $g^{x_3x_4+x_3x_1+x_4x_1}$ from u_1 . Calculate

$$\begin{aligned}
 K &= (g^{x_3+x_4+x_1})^{x_2} g^{x_3x_4+x_3x_1+x_4x_1} \\
 &= g^{x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4}
 \end{aligned} \tag{2.3}$$

For an eavesdropper to compute (2.3) from the publicly available information, he must obtain all the values of the form $g^{x_i x_k}$, $i < k$. For a given value of j (in our

Step	Operation performed by u_i
1	calculate g^{x_i} for random x_i and send to u_{i+1}
2	receive $g^{x_{i-1}}$ from u_{i-1}
	calculate $(g^{x_{i-1}})^{x_i}$ and send to u_{i+1}
$\vdots$	$\vdots$
t	receive $g^{x_{i+1} \cdots x_i x_1 \cdots x_{i-1}}$ from u_{i-1}
	calculate $K = (g^{x_{i+1} \cdots x_i x_1 \cdots x_{i-1}})^{x_i}$

Table 2.1: Conference key generation for Ingemarsson scheme.

example $j = 2$), it is necessary for an eavesdropper to “tap” $j + 1$ communication lines. Suppose an eavesdropper obtained messages going to u_i . He would be missing all values with the term x_i in the exponent. If he were also to observe messages going to u_k , he would observe terms with x_i in the exponent (and likewise terms with x_k in messages going to u_i) but would be missing the term with $x_i x_k$ in the exponent. Eavesdropping on one additional line would be required to recover $g^{x_i x_k}$.

In the above example, notice that eavesdropping on messages sent to u_2 reveals $g^{x_3 x_4 + x_1 x_3}$. Eavesdropping on u_3 would allow retrieval of $g^{x_1 x_2 + x_3 x_4 + x_1 x_4}$. Yet, an eavesdropper X would still be missing $g^{x_2 x_3}$. Observe that this term would not be sent to either u_2 or u_3 . Therefore, X would be required to eavesdrop on one additional line.

To obtain what [Inge82] calls Multitap Resistance (MTR), the protocol for producing K from the symmetric function $S^{(j)}(I)$ should have $j = |I| = t$ where t users are attempting to establish a key. In this way, an eavesdropper would be required to tap $t + 1$ lines to recover K . However, since there are only t lines to tap, this attack will fail.

Therefore, as mentioned earlier, for t users to establish a key K using this method, they will use (2.2). Table 2.1 lists the operations performed by u_i at each step of this computation. From this, we observe that each user calculates the key

$$K = g^{\prod_{i=1}^t x_i} \pmod{p}. \quad (2.4)$$

2.1.2 Burmester-Desmedt Scheme

In [Burm94], Burmester and Desmedt present a more practical alternative to Ingemarsson's symmetric function $S^{(i)}(I)$ (see Definition 2.1).

Definition 2.2 A cyclic function $C(t)$ is the sum of t products $x_i x_{i+1}$ where $i = 1, \dots, t$.

In other words,

$$C(t) = x_1 x_2 + x_2 x_3 + \dots + x_{t-1} x_t + x_t x_1.$$

Each user will compute the key

$$K = g^{x_1 x_2 + x_2 x_3 + \dots + x_t x_1} \pmod{p}. \quad (2.5)$$

The advantages of this scheme over the scheme of Section 2.1.1 are that

- the total number of interactions for the users is independent of the number of users and
- the security of the scheme is proven to be equivalent to the Diffie-Hellman scheme.

Recall that in Section 2.1.1, the number of rounds was equivalent to the number of users, i.e., t users required t rounds (see Table 2.1). As we will see, the present scheme requires only 2 rounds. The relevance and proof of the equivalence to Diffie-Hellman will be discussed later.

The key establishment proceeds as follows. Each user u_i

1. selects a random $x_i \in Z_p^*$ and broadcasts

$$y_i = g^{x_i} \pmod{p}. \quad (2.6)$$

2. Each user u_i then computes and broadcasts,

$$W_i \equiv \left(\frac{y_{i+1}}{y_{i-1}} \right) \pmod{p}.$$

3. Upon receiving the broadcasts from other users, computes the key,

$$\begin{aligned}
 K_i &\equiv (y_{i-1})^{x_i} W_i^{t-1} W_{i+1}^{t-2} \cdots W_{i-2} \pmod{p} & (2.7) \\
 &\equiv (y_{i-1})^{x_i} \\
 &\quad (y_{i-1})^{x_i} W_i \\
 &\quad (y_{i-1})^{x_i} W_i W_{i+1} \\
 &\quad \vdots \\
 &\quad (y_{i-1})^{x_i} W_i W_{i+1} \cdots W_{i-2} \pmod{p} \\
 &\equiv g^{x_1 x_2 + x_2 x_3 + \cdots + x_i x_1} \pmod{p} \\
 &\equiv K \quad , \text{ from (2.5) for all } 1 \leq i \leq t
 \end{aligned}$$

Notice that each u_i broadcasts only two messages. The first is y_i . (Actually, u_i only needs y_{i-1} and y_{i+1} .) The second is the combination of information from u_{i-1} and u_{i+1} ; W_i . This results in a drastic decrease in the amount of interaction, compared with the scheme from Section 2.1.1. Also notice that the exponentiations required to compute (2.7) are significantly less than in the computation of (2.4). In (2.4), the intermediate computations to produce the key, all involve exponentiating to an element in Z_{p-1}^* (see Table 2.1). Other than the computation of W_i , and the first part of (2.7), the amount of exponentiations u_i performs, are dependent upon t . For small conference groups, the calculation of (2.7) is much more efficient.

The following theorem illustrates that the present key agreement protocol is equivalent to the Diffie-Hellman key agreement protocol in a probabilistic sense. In other words, if an eavesdropper X 's ability to solve for (2.5) allowed X to find Diffie-Hellman keys, then solving for (2.5) is said to be at least as hard as solving Diffie-Hellman. If the reverse is also true (i.e., solving DH implies the ability to solve (2.5)), then the two problems are said to be equivalent. More specifically, an ability to solve for (2.5), allows computation of a Diffie-Hellman key with only a non-negligible probability. As a result of this lack of determinism, we say that the two protocols are probabilistically equivalent.

Theorem 2.1 *If t is even and polynomial bounded in the length of p then solving for the key K from (2.5) is probabilistically equivalent to solving a key for the Diffie-Hellman protocol (see Section 1.1).*

Proof:

We first show how an eavesdropper X would use the ability to solve for a Diffie-Hellman key, to allow solving (2.5).¹ Given g , $g^x \bmod p$ and $g^y \bmod p$ for some $x, y \in Z_{p-1}^*$, evidence for an ability to solve for a Diffie-Hellman key would be the computation of $g^{xy} \bmod p$, i.e. $g^{xy} \bmod p = DH_p(g, g^x, g^y)$ for a Diffie-Hellman oracle DH_p . Given the y_i 's of each user from (2.6), X can compute $v_i = DH_p(g, y_i, y_{i+1}) = g^{x_i x_{i+1}} \bmod p$ for $i = 1, \dots, t$. From this, X can compute

$$K \equiv v_1 v_2 \cdots v_t \pmod{p}$$

which is equivalent to the session key from (2.5).

We now show how to use an ability to solve for (2.5) to solve for an instance of a Diffie-Hellman key involving y_1 and y_t , i.e., solve for $g^{x_1 x_t}$. Given y_1 and y_t , we prepare the remaining input for X . Select $b_2, \dots, b_{t-1}$ at random from Z_{p-1}^* and compute

$$\begin{aligned} y_2 &\equiv y_1 g^{b_2} \pmod{p} \\ y_3 &\equiv y_2 g^{b_3} \pmod{p} \\ &\vdots \\ y_{t-1} &\equiv y_{t-2} g^{b_{t-1}} \pmod{p} \end{aligned}$$

It follows that $W_1, \dots, W_{t-2}$ can be computed efficiently as

$$\begin{aligned} W_i &\equiv \left(\frac{y_{i+1}}{y_{i-1}} \right)^{x_i} \pmod{p} \\ &\equiv (g^{b_{i+1}})^{x_i} \pmod{p} \\ &\equiv (g^{x_i})^{b_{i+1}} \pmod{p} \\ &\equiv (y_i)^{b_{i+1}} \pmod{p} \end{aligned}$$

¹This portion of the proof was omitted from [Burm94].

We now use the fact that t is even, to produce

$$\begin{aligned} y_t &\equiv y_2 g^{-b_2} \equiv y_4 g^{-b_2-b_4} \equiv \dots \equiv y_{t-2} g^{-b_2-b_4-\dots-b_{t-2}} \pmod{p} \\ y_1 &\equiv y_3 g^{-b_3} \equiv y_5 g^{-b_3-b_5} \equiv \dots \equiv y_{t-2} g^{-b_3-b_5-\dots-b_{t-1}} \pmod{p}, \end{aligned}$$

allowing us to compute

$$\begin{aligned} W_{t-1} &\equiv \left(\frac{y_t}{y_{t-2}} \right)^{x_{t-1}} \equiv (y_{t-1})^{-b_2-b_4-\dots-b_{t-2}} \pmod{p}, \text{ and} \\ W_t &\equiv \left(\frac{y_1}{y_{t-1}} \right)^{x_t} \equiv (y_t)^{-b_3-b_5-\dots-b_{t-1}} \pmod{p} \end{aligned}$$

Using the information available to him and by our assumption, X is now able to compute K_1 , i.e., the key computed by u_1 . From K_1 (see (2.7)), X can obtain $(y_t)^{tx_1} \equiv (g)^{tx_1 x_1}$. From [Rab80], n th-roots modulo a prime p can be computed efficiently when t is polynomially bounded. Of the possible roots computed, X would randomly select one as the correct Diffie-Hellman key. Therefore, with a greater than a non-negligible probability (i.e., $\frac{1}{p}$), X will select the correct K . This shows that it is feasible to solve the Diffie-Hellman problem, given an ability to solve for (2.5), contradicting the assumption that it is hard to solve Diffie-Hellman. $\square$

2.1.3 Steer Scheme

In [Stee88], a protocol was presented to satisfy the requirements for a secure audio teleconference system. For our purposes, we merely examine how the set P can securely and mutually obtain a conference key K . Initially, each user u_i chooses a random x_i and computes $y_i = g^{x_i}$. These y_i are then broadcast to all other users. Once this is complete, the protocol for obtaining a conference key K consists of different computations for each of the users, i.e., it is not symmetric.

Each user will compute the following key:

$$K = g^{x_1 g^{x_{t-1}} \dots g^{x_3 g^{x_2 x_1}}} \pmod{p} \quad (2.8)$$

Since the order of operations for exponentiation is right to left, the calculation of K must begin with u_1 . Given the y_j of all other users, u_1 proceeds to compute the following:


```

 $K = x_1$ 
for  $j = 2$  to  $n$ 
     $K = y_j^K$ 

```

The remaining users (except for u_2) cannot compute K given only their secret information, and the y_j of all other users. Notice from (2.8) that u_2 can compute the key in the same way as u_1 . However, u_3 requires knowledge of $g^{x_1x_2}$ to compute K . The same is true of the remaining users. The position of u_j 's x_j in (2.8) is important in that u_j can compute "to the left" (since exponentiation is computed right to left) but needs to have information "from the right" (from u_{j-1}) to start off his computation of K . For example, u_3 is given $g^{x_1x_2}$ from u_2 , which allows u_3 to compute K . To compute K , each u_j , for $j = 2, \dots, t-1$ proceeds to

- receive w_{j-1} from u_{j-1} (u_2 uses $w_1 = y_1$)
- Calculate $K = w_{j-1}^{x_j}$ and $w_j = g^K$.
- Send w_j to w_{j+1} and compute

```

for  $c = (j+1)$  to  $n$ 
     $K = y_c^K$ 

```

Since u_t does not have to help anyone with the computation of K , u_t simply receives w_{t-1} from u_{t-1} and computes $K = w_{t-1}^{x_t}$.

2.1.4 Comparison of Schemes

The most important point to make is that these schemes do not capture the simplicity of the basic Diffie-Hellman protocol. In that model, each user need only broadcast one piece of information, dependent only upon a secret key of their choosing and a set of public parameters. However, when we attempt to extend this to more than one user, the models we have presented show us that the process is much more complicated.

The effect that these extensions introduce is an increased amount of interaction for all three models, and a non-symmetric protocol for the Steer model. The best attempt at achieving a Diffie-Hellman multi-party key agreement scheme was from the scheme in Section 2.1.2 by Burmester and Desmedt (BD). It drastically improves the computation and interaction from the Ingermarsson scheme (ING) in Section 2.1.1. The scheme from Steer *et al.* Section 2.1.3 (STR) reduces the broadcasts of each user but maintains the same number of overall rounds required to compute the key as in ING. We say that a new round is initiated each time any user broadcasts an additional piece of information to aid another user in his key calculation.

Another important point to make is that the lack of a strict broadcast system for these schemes requires that each user know his "position" or "numbering" in the computation of the key. Presumably, this will be done at set-up, or there may be some implied method for users to determine their relative positions.

For a group of t users to establish a key K , ING required t steps or rounds. Each user adds his secret piece to a message he receives, and passes this on to the next user. STR also requires t rounds. Even though u_1 can calculate K immediately after the first round, u_t must wait until $t - 1$ rounds have passed before he has received enough information to compute the key. Therefore, t rounds are required for *all* users to compute the key. On the other hand, BD requires only 2 rounds to complete its key computation. The first round is the initial broadcast of the y_i 's. The second is the broadcast of the W_i 's. Although both STR and BD require only 2 broadcasts by each user, the lack of symmetry in STR results in the need for more rounds to be completed before the entire protocol is done.

The computation of the session key is similar in both ING and STR. Each user performs t modular exponentiations in the computation of K . In ING, each user is simply adding his own exponent to a partial key computation and passing the result to the next user. The last round requires only 1 exponentiation, i.e., with the user's own secret piece. In STR, u_i performs $t - i + 1$ exponentiations. In other words, u_1 performs t modular exponentiations, while u_t performs 1. On average, each user will perform approximately t modular exponentiations. In BD, each user

	ING	BD	STR
number of rounds	t	2	t
number of broadcasts	t	2	2
key computation	$O(t \lg p)$	$O(\lg p) + O(t \lg t)$	$O(t \lg p)$

Table 2.2: Key Agreement Comparison

also performs t exponentiations. It is in the size of these exponentiations that a computational reduction is made. In ING and STR each modular exponentiation will require at most $\lceil 2 \lg p \rceil$ modular multiplications. From (2.7), notice that only one of the exponentiations requires at most $\lceil 2 \lg p \rceil$ modular multiplications. The remaining exponentiations require at most $\lceil 2 \lg(t - j) \rceil$ modular multiplications for W_j (see (2.7)). This results in

$$2 \sum_{j=1}^t \lg(t - j) = 2 \sum_{k=1}^{t-1} \lg k = 2 \lg \prod_{k=1}^{t-1} k = O(t \lg t)$$

additional modular multiplications. These results are summarized in Table 2.2.

Therefore, as long as $t \ll p$ the key computation of *BD* will be much more efficient. For practical applications, p will be an integer on the order of 10^{50} . Considering that there are only 5×10^9 humans living on the earth, we will always have the situation where $t < p$.

2.2 Associative One-Way Functions

In [Rabi93], Rabi and Sherman introduce the concept of an *associative one-way function* (AOWF) and discuss their application in solving the key agreement problem. Despite the lack of a concrete implementation, they present a strong foundation for the building of such applications. As we will see, if candidates for strong, commutative AOWFs are found, we can construct a multi-party key agreement protocol that captures the elegance of the two-party Diffie-Hellman protocol.

Not suprisingly, an associative one-way function, is a binary function that is associative and one-way. Let $S = \{0, 1\}^*$ be the set of all binary strings. A function

$\odot : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ is *one-way*, if it is easy to compute $z = x \odot y$ for all $x, y \in \mathcal{S}$ yet given $z \in \mathcal{S}$, it is hard to compute any x and y satisfying $z = x \odot y$, for most z , i.e., it is hard to *invert* $\odot$. $\odot$ is *associative* if $(w \odot x) \odot y = w \odot (x \odot y)$ for all $w, x, y \in \mathcal{S}$.

An AOWF $\odot$ is said to be *strong*, if given x and $z = x \odot y$, it is hard to recover y (likewise, if given y instead of x). In [Rabi93], it was shown that AOWFs exist iff $P \neq NP$. According to [Sher94], it was later shown that strong AOWFs exist iff $P \neq NP$.

The distinction between regular and strong AOWFs is important to the implementation of a multi-user key agreement protocol. This is evidenced by the fact that no strong AOWFs are known. However, consider the following examples of AOWFs identified in [Rabi93]²:

integer multiplication Multiplication of large odd prime numbers is a provable AOWF [Rabi93]. It is associative (i.e., $(pq)r = p(qr)$), and the operation of factoring the product of two large primes is believed to be extremely difficult.

In fact, several cryptosystems rely on this being true (i.e., [Rabi78],[RSA78]).

matrix multiplication Analogous to integer multiplication.

Although there are examples of AOWFs, there are no known strong AOWFs. Notice that integer multiplication is not a strong AOWF. Given p and $n = pq$, one can easily determine q . Despite the lack of a strong AOWF, the following section illustrates how AOWFs were used by Rabi and Sherman [Rabi93] to obtain a multi-party key agreement protocol.

2.2.1 Using AOWFs for Multi-Party Key Agreement

To use AOWFs for multi-party key agreement, we require that they be strong, and commutative. First suppose that there is a public parameter $y \in \mathcal{S}$ that is known to all users. The function $\odot$ is also known to all users, and computable in polynomial time.

²The logical OR of two n -bit binary strings was also presented as an example in [Rabi93]. However, as acknowledged in [Rabi93], it does not suitably fit the definition of an AOWF.

Each user u_i selects a random $x_i \in S$. The secrecy of x_i is maintained, while $x_i \oslash y$ is broadcast to all users. Notice that an eavesdropper X , has possession of y and $x_i \oslash y$. However, because of the assumptions made for a strong AOWF, X is not able to recover x_i . User u_i , upon receiving the broadcasted information from the other $t - 1$ users, computes the key

$$\begin{aligned} K_i &= x_i \oslash (x_1 \oslash y) \oslash \cdots \oslash (x_{i-1} \oslash y) \oslash (x_{i+1} \oslash y) \oslash \cdots \oslash (x_t \oslash y) \\ &= x_1 \oslash \cdots \oslash x_t \oslash y^{t-1} \end{aligned}$$

Because of the commutativity and associativity of $\oslash$, each user will compute the same key.

Chapter 3

Secret Sharing

The concept of secret sharing was independently developed by Shamir [Sham79] and Blakley [Blak79]. It was proposed as a method of key management. An entity shares a secret with users in such a way that only specified subsets of users can recover the secret. We begin by illustrating how the need for secret sharing arose.

Consider the situation where some information is protected by a “lock”, and only one person has the “key” to open this lock. For example, consider a bank vault that only the bank’s manager can open. Some of the obvious problems with this system are

- loss of key
- key is forgotten (i.e., for a combination lock)
- sabotage (key is destroyed)
- requires 100% confidence in holder of key

The last point is really a general property of the first three, and emphasizes that for this scheme, we have “placed all of our eggs in one basket”.

For anyone who has locked themselves out of their home or car, the first two items will sound familiar. A simple solution to this problem is to keep multiple copies of the key. In this way if the key is lost, forgotten or even destroyed, there are other keys that can be used to open the lock.

For this convenience, we pay the price of having to trust all of the individuals possessing a copy of the key (or trust that we have securely and safely stored each copy). If only one of the key holders is corrupt, the security of the system has been compromised.

The single and multiple key solutions illustrate the extremes to this problem. The tradeoffs between the two can be summarized with the following.

security vs. reliability It is unreliable to have one key and security is compromised when there are many copies.

security vs. convenience of use It is convenient for many people to have the key, but obviously not very secure.

The second point is worthy of some further explanation. Consider the payroll department of your favorite company. Suppose that the opening of a “lock” is the ability of a person to sign a paycheck. The success of a person to digitally sign a cheque identifies a person with a valid key. If many people are given this ability, the system is convenient, yet very insecure. Yet if only one person were given this ability, any of the previous problems illustrated with such a one-key system would be clearly unacceptable, i.e., people cannot be paid if the cheque-signer is absent.

In the following section, we present a method that combines the convenience of having a number of copies of the key with the security of having only one copy. We follow this by describing the schemes of the independent co-inventors of this method; namely Shamir [Sham79] and Blakley [Blak79]. In the final section of this chapter, we review methods of including reusability in threshold schemes.

3.1 Threshold Schemes

Informally, a threshold scheme allows a key to be divided into a number of “pieces”, such that a specified number of these pieces (the threshold level), are required to recover the key. It is a method of key management in the sense that the risk of key loss or compromise is lessened. Security is achieved by the fact that groups of

Figure 3.1: Given a key K , the concealer function produces a share s_i for each of the n users.

users must be formed before an action is taken. More importantly for our purposes, threshold schemes are also a method of key establishment in the sense that once the threshold is reached, a key has been established.

More formally, a (t, n) *threshold scheme* implies a method for sharing a secret key K among a finite set U of n users such that a subset P of at least $t \leq n$ users from U can recover K while no group of size less than t can do so (see [Blak80, Sham79, Simm91]). A trusted center T randomly selects a key K and uses a *concealer function* to produce n shares or *shadows* on input K . This process is illustrated in Figure 3.1.

The n shares are then securely distributed to the n users, each user receiving exactly one share. To recover K , the users in the privileged set P , where $|P| \geq t$, combine their shares and input them to a *revealer function* to produce K . Figure 3.2 illustrates the case where exactly t shares are input to the revealer.

For every set $Z \subset U$, where $|Z| < t$, if inputting less than t shares to the revealer function will give the users in Z no advantage in obtaining K , the threshold scheme is said to be *perfect*. By no advantage, we mean that if less than t users were to input their shares to the revealer function, they would not have a greater probability of obtaining K than a user who only had his own single share.

To obtain a perfect secret sharing scheme, the size of the share for each user, must be at least as large as the size of the key K . Suppose that this were not true, i.e., for some s_j , $|s_j| < |K|$. As an illustrative example, suppose that K is 100 bits in length, while s_j is 99 bits long. Let R represent a set of $t - 1$ users (not including user u_j) who input their shares to the revealer function. On their own, these users have 100 bits of uncertainty in K . However, since they only require 1 additional

Figure 3.2: Given the shares of at least t users, the revealer function produces the corresponding key K .

share to be input to the revealer function, knowledge of s_i would allow recovery of K . Since s_i is only 99 bits long, the set R of users have an advantage in recovering K , by guessing s_i rather than K . Notice that if u_i had his share alone, he would have 100 bits of uncertainty in K . Hence, in a perfect secret sharing scheme, it is necessary that each user's share is at least as large as K . When there is equality, the scheme is said to be *ideal*.

The following two simple threshold schemes illustrate the extremes for threshold schemes. In item 1, each user is given a copy of the key. This is a convenient and reliable scheme, though very insecure, as mentioned on page 26. The second is very secure, though inconvenient since the pieces of all users are required to initiate an action, i.e., open a lock.

1. $(1, n)$. In this scheme, the threshold for obtaining K is 1. Each share s_i is assigned the value K from the concealer function (see Figure 3.1). This is exactly the scheme mentioned previously where a number of copies of the key are kept.
2. (n, n) . In this scheme, all of the shares are required to recover the key. Consider a 3-number combination lock where each of three users are assigned one of the numbers.

3.2 Shamir's Threshold Scheme

Shamir [Sham79] proposes a method for implementing a (t, n) threshold scheme, based on *polynomial interpolation*. The scheme relies on the fact that t points are required to uniquely identify a polynomial of degree $(t - 1)$.

The trusted center T will perform the following steps to set up the system.

1. Randomly select a key $K \in Z_p^*$ for a prime $p > n$.
2. Randomly select $t - 1$ distinct $a_i \in Z_p^*$
3. Randomly select n distinct $x_i \in Z_p^*$ (the x_i can be publicized). Each x_i is assigned to u_i .
4. Let $f(x) = K + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1}$ and compute $y_i = f(x_i), 1 \leq i \leq n$.
5. Distribute y_i to u_i over a secure channel.

At the end of this set-up stage, each user possesses a point on the polynomial $f(x)$. Whether or not x_i is published, the corresponding y_i is known only to u_i . Recovery of K requires t users to input their shares to the revealer function (see Figure 3.2). In this specific case, the revealer function is *polynomial interpolation*. Recall that t distinct points are required to recover a polynomial of degree $t - 1$. Such is the case we have here, where $f(x)$ is a $t - 1$ degree polynomial.

The polynomial $f(x)$ is recovered using the Lagrange Polynomial

$$f(x) = \sum_{i=1}^t \frac{y_i(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_t)}{(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_t)} \pmod{p}. \quad (3.1)$$

However, if we notice that K is simply the coefficient of $f(x)$, we have

$$K = f(0) = \sum_{i=1}^t \frac{y_i(-1)^t(x_1) \cdots (x_{i-1})(x_{i+1}) \cdots (x_t)}{(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_t)}. \quad (3.2)$$

Therefore, to recover K , t users must use their shares to compute (3.2).

Shamir's threshold scheme is perfect in the sense that the combination of $t - 1$ shares reveals no additional information about the key K (i.e., since the size of a

share is equal to the size of K , guessing a valid additional share or K itself have the same probability of success). Even if a valid x -coordinate were known (as described below), (3.2) requires an additional y_i to complete the equation. The problem being that for any y_i one might guess, (3.2) will produce a valid key (though only one should produce the correct key). Therefore, guessing a y -component of a share is no more advantageous than guessing the key.

If the x -coordinate of each user's share is made public, then Shamir's scheme is ideal. Notice that each user's share consists only of the corresponding y -coordinate. Since the y_i 's and the key itself are each chosen from Z_p^* , the size of each share is equal to the size of the key. However, this arrangement introduces another problem. Having recovered the secret polynomial $f(x)$, u_i can determine u_j 's secret share, given the public x_j , i.e. $y_j = f(x_j)$. This problem is discussed in Section 3.4.

3.3 Blakley's Threshold Scheme

As a second example of a threshold scheme, we describe the method introduced by Blakley (the independent co-creator of threshold schemes) in [Blak79]. The scheme is similar to Shamir's in that it is based on linear algebraic concepts. One difference is that Blakley prefers to use the term "shadows" rather than "shares" when referring to the secret pieces of information securely distributed to each user. Similarly, for a (t, n) threshold scheme, t shadows are required to recover the key. However, unlike Shamir's scheme, Blakley's scheme is not "perfect".

Blakley's threshold scheme is far less intuitive than the interpolation counterpart presented by Shamir. For this reason, we only describe a scaled down $(2, 3)$ version of Blakley's scheme, similar to the one given by Simmons [Simm91].

The key K is a point p in 3-dimensional space (see Figure 3.3).¹ In Shamir's scheme, K was the intersection of the $t - 1$ degree polynomial $f(x)$, with the y -axis.

¹Although a 2-dimensional space would have been sufficient to describe this simple generalization of Blakley's scheme, it would not have illustrated the fact that Blakley's scheme is not perfect.

Figure 3.3: Blakley's Threshold Scheme

In Blakley's scheme, the key is the intersection of the shadows given to each user. These shadows, are lines projected from either of the three axes, to the point p . For example, the line l_x in Figure 3.3 is a projection from the yz -plane, along the x -axis to the point p .

This specific example is a $(2,3)$ threshold scheme. Notice that only two non-parallel lines are required to identify p from Figure 3.3. A user in possession of one of the appropriate lines, has an advantage in obtaining p (hence Blakley's scheme is not perfect). Suppose a user possesses the line l_x from Figure 3.3. This user need only guess the appropriate z -coordinate since knowledge of l_x has fixed the x and y coordinates. A user not in possession of an appropriate line must guess all three of the coordinates.

However, it is important to note that each guess of a z -coordinate for p will produce a valid (though most likely incorrect) key. Also notice that a user in possession of an appropriate share, i.e., l_x , has the same uncertainty in the key p as a user of Shamir's scheme. (The z -coordinate in Blakley's and the y -intercept in Shamir's are each elements of Z_p^* .) Therefore, saying that Blakley's scheme is not perfect, while Shamir's is, does not indicate a lack of security in the former as compared to the

latter. The distinction is merely based on the definition of the word “perfect”, as a relative comparison of share size to key size.

3.4 Reusability of Threshold Schemes

As a result of the interactive nature of threshold schemes, the loss of a user’s share has farther reaching implications than simply a loss of the share for one user as in a normal cryptosystem (i.e. since all the users in a threshold scheme recover the same key, the loss of a single share affects other users, while in a normal cryptosystem, the loss of a user’s key only affects that user since it only allows decryption of messages sent to that specific user). Consider that an outsider (a person who was not given a share by T) who obtains the share now has an advantage over an outsider that does not possess such a key. The advantage being that this outsider (like all legitimate users) has only a threshold of $t - 1$. Note how this differs from our previous use of the term advantage where a user would have an advantage if guessing the t^{th} share input to the revealer function was “easier” than simply guessing the key. Here, we are considering an active, malicious adversary whose goal is to steal shares (if his goal were to simply guess shares, he would be better off guessing the key in most cases). Clearly, each additional share brings him closer to his goal, hence he has an increased advantage from when he started.

This type of compromise was termed a *disenrollment* by Blakley [Blak92]. The problem was to maintain the information theoretic secrecy of the key K (to be established by a privileged set of users), despite any disenrollments. The obvious solution is to distribute new shares to users over a secure channel, once a disenrollment has occurred. These new shares, if independent from the old ones, essentially initiate a new threshold scheme whereby the disenrolled share is of no value (the same can be said for all of the original shares of the remaining users). However, the need for a secure channel negates the appeal of the scheme.

The solution in [Blak92] allows for new keys to be distributed over an insecure channel, while maintaining the same threshold value. To be able to handle L disen-

rollments (i.e., L key compromises) and maintain a perfect threshold scheme, each user must have a share that is $L + 1$ times the size of the key K (this bound is shown to be optimal in [Blak92]). To achieve such a scheme, the following setup can be employed. Let user u_i possess $L + 1$ random, independent integers $S_0, R_1, \dots, R_L$, that will be used as *one-time pads* (see [Denn83]) for distributing new shares. Initially, S_0 is the share of u_i . Once a disenrollment occurs, u_i is issued a new share S_1 that is independent of S_0 . This share will be sent to u_i from T as the message $R_1 \oplus S_1$, where $\oplus$ is bitwise exclusive or and R_1 is used as the one-time pad on the new share S_1 . Only u_i can recover S_1 given that R_1 and S_1 are random and independent, i.e., $S_1 = (R_1 \oplus S_1) \oplus R_1$.

Other than disenrollments, a possible concern for secret sharing schemes is the publication of part of a user's share. This can lead to a disenrollment if the publication of one part of a user's share compromises the remainder of the share. This is evident in Shamir's scheme where the publication of the x -components introduces the possibility of one user to recover the shares of other users.

There are two possible solutions to the problem described above. The first is to ignore it. Suppose that our threshold scheme was used to allow entry to a locked room, only if t out of n users are present. Each of the n users are given a card with their share on the card (the card is created such that the shares are not known nor accessible by the users). To open the door, t users must pass their cards through a card reader located on the door. In this way, the process can be repeated a number of times, as nothing is "revealed" to the users; the t users only see the effect of using the key. A second example is with the launching of missiles. In its simplest form, a $(2, 2)$ threshold scheme is used, where 2 users, with proper keys, are required to launch a missile. Therefore, there are applications where the fact that users shares might otherwise be compromised, can be ignored.

For applications where the publication is a concern, the following technique can be employed. Once the key has been revealed, users can potentially determine the shares of other users. One can think of the share compromises as a large disenrollment. The difference being that a number of shares have been compromised,

rather than just one. But consider that this really is the same, since when a single disenrollment occurs, the loss of one share also invalidated the remaining shares. Therefore, we could solve the problem by using the same technique presented above for disenrollments.

A more practical solution is presented by Zheng *et al.* in [Zhen94]. They make use of pseudo-random function families and universal hash function families. A pseudo-random function family is simply a collection of pseudo-random functions (which are defined later), each having a specific index for defining the function in the family (the same applies to universal hash function families).

Each user u_i chooses his own random share s_i . When t users intend to combine their shares to produce a key K (chosen *a priori* by the center T), they first compute the value of a pseudo-random function (PRF) f , on some designated identification for that user, keyed by their share s_i , i.e., u_i computes $z_i = f_{s_i}(I_i)$. (Besides personal information for u_i , I_i also includes a publicized identification N_K , pertaining to the key K .) A PRF f can simply be thought of as a function that produces an output that appears to be totally unrelated to its input. The t users would then combine their z_i to compute X_j as follows:

$$X_j = z_1 \oplus z_2 \oplus \cdots \oplus z_t,$$

where $\oplus$ is the bit-wise exclusive or, and we assume without loss of generality that $u_1, \dots, u_t$ represent the users establishing K .

X_j is such that for each possible X_j that a subset of t users could create, $h(X_1) = \cdots = h(X_w) = K$ for a hash function h . For t users, there are $w = \binom{n}{t}$ possible subsets of t users. Such a hash function h is called *w-collision accessible* [Wegm81]. In other words, w X_j 's can be found such that they each map to K . The specific hash function h can be publicized.

A trusted center T is required to choose and publish the functions f and h . This involves some initial interactions between T and the users. These interactions are necessary to allow T to find a function h satisfying $h(X_1) = \cdots = h(X_w) = K$. Each

time a key K has been revealed, a new hash function h' is published. This must be done or else each set of t users would recover the same key K each time.

Therefore, users are able to keep their shares secret because they are using the output of a PRF f to produce K . Although other users may be able to recover $f_{s_i}(I_i)$ in the computation of K , they are unable to recover s_i . Since a new hash function is used for each key establishment, the same set of users will not retrieve the same key during a new key establishment. Also, since a new key is used for each establishment, the identification number N_K for each key will also be different. Therefore, $f_{s_i}(I_i)$ will differ for each key establishment. For further description of the algorithm and for an implementation of the scheme, consult [Zhen94].

Chapter 4

Broadcasting Session Keys

Consider the situation where we have a center that wishes to share a message with a subset of a larger set of users. The users have registered with the center *a priori*. We will therefore assume that the center has some means of uniquely identifying each user. There are two basic ways for the center to carry out this exchange.

The first is for the center to address the message individually for each user in the subset. This method is known as a *point-to-point* exchange. It requires that each message be individually addressed to the desired users, thereby requiring a larger amount of information to be sent out. If the subset is of size t , then an additional $\lg t$ bits must be appended to each message.

Since each user is receiving the same message, it would seem more logical for the center to have to send out only one copy of the message. This method is known as a *broadcast* or *point-to-multipoint* exchange of the message (see [Gopa82]). In such an arrangement, those users who share a communication line with the center, will receive the message. This is because the message is broadcast from the center, i.e., sent on all outgoing lines. For example, consider a local cable television company.

To incorporate the fact that the center would like to exchange a message with only a subset of these users, we have the example of pay-television. Even though all users have a communication line connected to their homes, only those users with a proper decoder can recover the intended message (e.g., movie).

In the techniques described above, the only concern was with a simple exchange

of a message. To prevent against potential eavesdropping of the communication line, encryption must be employed. However, the direct use of encryption introduces some problems.

Consider a system where we have a set U of n users who are registered with a *trusted center* T . Upon registration, T will securely share a unique secret s_i with each user u_i . This s_i will be a long-term *cryptographic key*. The subset of users that T wishes to exchange a message with will be the privileged set P of size $t \leq n$.

For a point-to-point encrypted exchange, not only must the address of the user be appended to each message, but the message must be individually encrypted for each of these users. For an integer message M , T must perform t encryptions of M and distribute $t(\lg M + \lg t)$ bits. For a large message M , this can be very costly with respect to both time and space.

Since our goal is to achieve a secure exchange of information with a group of users, T could *a priori* share distinct keys with groups of users, rather than a different key for each user. This solution is useful if T knows the groups that he will exchange information with beforehand. However, in many situations, group membership is more dynamic. For this situation, T would have to anticipate a larger number of possible groups. This would require each user to store a large number of keys, proportional to the number of groups to which they belong.

Rather than sharing keys with groups of users beforehand, T will first distribute a *session key* K to the members in P . This is a dynamic group key in that it will only be used for the message exchange(s) with the current members in P . Each u_i will continue to share a single key s_i with T . K will be encrypted and addressed individually for each user in P , while saving a large amount of work for T since K will usually be much smaller than M . This solution also has the advantage that subsequent messages can be exchanged with P , encrypted under the same K .

Notice that this solution also allows M to be broadcast. Since M will only be encrypted under K , those users who have K can recover M from the encrypted message. Only users in P have this knowledge. However, we have retained the problem of having to address and separately encrypt K for each user.

Figure 4.1: Serial encryption of a session key K

In this chapter, we examine methods of broadcasting session keys. Section 4.1 reviews methods of broadcasting that perform a serial encryption of the session key. In Section 4.2, we introduce a new framework that allows for parallel encryptions of session keys. Section 4.3 presents a lower bound on the size of the broadcast message for a subset of schemes satisfying the model of Section 4.2. In Section 4.4 we present a method for using threshold schemes from Chapter 3 to create a parallel broadcast scheme. We improve one scheme such that it achieves the lower bound from Section 4.3. Finally, Section 4.5 shows how to use two computationally intractable number theoretic problems to create secure parallel broadcast schemes.

4.1 Serial Broadcasting

In this section, we present some of the previous attempts found in the literature that solve the problem of securely broadcasting a session key K . The method used in each is similar in that the same basic construction of the broadcast message is used.

The session key K is first encrypted individually for each of the specified users, using their secret s_i . These separate encryptions are then combined to form the larger broadcast message C (see Figure 4.1). Decryption involves recovering the encrypted piece from the broadcast message C , and decrypting it with s_i to recover K . Section 4.1.1 and Section 4.1.3 present methods of constructing C from the individual encryptions.

4.1.1 The Secure Lock

In [Chio89], the authors propose a method for one user u_B to broadcast a session key K , to a privileged subset P of users, using what they call a *secure lock*. The process is such that only one ciphertext need be broadcast by u_B , where only members in P can recover K from this message.

The procedure for creating the broadcast message is built on top of an existing cryptosystem. For this reason, the authors claim that their broadcasting scheme is as strong as the underlying cryptosystem. To be more specific, suppose that the set U of users (including the user u_B) are able to exchange messages via a public-key cryptosystem (PKCS), with function E (see Section 1.1). For simplicity, we assume that E is used to both encrypt and decrypt. Each user u_i has a private key d_i and publicizes a key e_i . To send a private message M to u_i , user u_B (or any other user) computes $C = E_{e_i}(M)$ and send C over an insecure channel to u_i . Since u_i has maintained the secrecy of d_i , no one else should be able to compute $M = E_{d_i}(C)$.

To allow u_B to broadcast a session key K , [Chio89] use the Chinese Remainder Theorem (CRT) (see Appendix A.2), and proceed as follows. Suppose we have a privileged set P of size t . Along with the requirements for the PKCS stated above, each u_i will also have a publicized integer n_i such that $\gcd(n_i, n_j) = 1, i \neq j$.

To privately share K with members in P , u_B will first compute $r_i = E_{e_i}(K)$, for

$u_i \in P$. For the following series of equations,

$$\begin{aligned} C &\equiv r_1 \pmod{n_1} \\ C &\equiv r_2 \pmod{n_2} \\ &\vdots \\ C &\equiv r_t \pmod{n_t} \end{aligned} \tag{4.1}$$

u_B can solve for a unique C by implementing the proof of Theorem A.1 (CRT). Upon receipt of C , user $u_i \in P$ will compute $r_i = C \bmod n_i$. Following this is computation of $K = E_{d_i}(C)$, thereby allowing retrieval of the session key. The CRT is used to create a single broadcast message, allowing for the symmetric recovery of a user's encrypted piece from this message, i.e., each user performs the same operation on C (albeit with their own n_i) to recover their piece r_i .

Before continuing, we note that the same process can be undertaken for users in a private-key cryptosystem. Rather than having corresponding public and private keys, each user u_i will share a single key e_{ij} with every other user u_j in the system. The same steps will be taken by u_B to create the broadcast message, with the addition that an indication of who created the broadcast message (u_B in this case) must be included. If r_i is obtained from the sealed lock by user $u_i \in P$, where $r_i = E_{e_{Bi}}(K)$, then u_i must be able to determine who created r_i , to allow him to use the correct e_{Bi} to recover K , i.e., he must know the B in e_{Bi} . Recall that u_i does not have just one deciphering key as in a PKCS, but shares a distinct key with every other user in the system.

The authors point out the following advantages to this scheme:

1. Only one copy of the ciphertext is sent out.
2. No extra keys are needed for each user in the system.

The first item is evident from the fact that u_B will only broadcast the message C . Since we are using a point-to-multipoint exchange, no addressing of the message is required. Only those members of P can recover the correct session key K . Included

in [Chio89] is the fact that the broadcast message can also contain the message $V = E_{K'}(K)$ where K' is the public portion of the session key K . In other words, only if a user has obtained the correct K from C , will he also obtain $K = E_{K'}(C)$. In this way, users are able to identify whether or not they were intended to receive a session key from u_B .

The motivation for stating the second point is an implication of the fact that the broadcast scheme is built on top of an existing cryptosystem. Each user requires only 1 key for the cryptosystem (or a key pair for a PKCS). The use of the CRT provides no extra security for the system but merely allows for computation of the broadcast message. For this reason, the n_i assigned to each user are not actual keys in the cryptographic sense. In fact, since the n_i are publicized, anyone can open the lock C to recover the r_i of any user in P . The main purpose in creating the broadcast message is to avoid the need to address individual messages to users in P . Therefore, the n_i are not considered keys.

As mentioned earlier, the authors claim that this system is as strong as the underlying cryptosystem upon which it is built. The authors present several possible attacks to their system and propose that the security will not degrade, so long as the underlying cryptosystem can withstand a *known-plaintext attack* (KPA) (see Section 1.1). We first note that if a PKCS is being used, the requirement to withstand a KPA is clear (more specifically, a PKCS is susceptible to a chosen plaintext attack). However, we point out that the context in which they claim security based on resistance to a KPA in the first attack is misguided.

The attack is by a cryptanalyst X , who is not a member of P , and is attempting to obtain the session key K from the sealed lock C . (We assume here that the KPA attack is based on X 's knowledge of a known session key, from a previous session, although this point was not made in [Chio89].) Recall that in a KPA, X is given a number of distinct plaintext-ciphertext pairs encrypted under the same key and is to attempt to infer either the key or a decryption of the next observed ciphertext. However in this broadcast system, X has a number of plaintext-ciphertext pairs for the same plaintext, encrypted under different keys (from a previous session

for which X has knowledge of the key K'). (Recall that since the n_i are public, anyone can recover the encrypted pieces intended for users in P .) From this, X is to attempt to infer the current session key K . Note the difference between the their claim, and a KPA.

In fact, in Section 1.2, it was shown how the RSA cryptosystem can be misused in such a situation. Recall that a cryptanalyst X was able to recover the message M (in this case a key K) using only a ciphertext only attack, if the message was encrypted for multiple recipients. This occurred despite the fact that RSA (using the same setup) is believed to withstand even a chosen plaintext attack, when sending a message to a single recipient. Therefore, it is incorrect for the authors to claim that a cryptosystem secure against a KPA, is also secure when creating multiple encryptions of the same message.

Another important point to notice is that this scheme does not solve all of the problems that an encrypted point-to-multipoint exchange introduced. Consider that for many security applications, it is prudent to change the session key so that the same key is not used for all message encryptions. In this way, compromise of the session key should not allow a cryptanalyst to decrypt all of the information, i.e., it only allows decryption of material encrypted with the specific encryption key. Updating the session key in this scheme requires re-encrypting K for each user, followed by computing a new lock C . We would like to have a scheme that avoids both the individual addressing and the individual encryptions. In Section 4.2, we present a scheme that solves this problem.

4.1.2 Using the Secure Lock

In [Chan92], a broadcast scheme using the secure lock from Section 4.1.1 is presented. The ElGamal public-key cryptosystem (PKCS) is used for encrypting the session key K [Elga85].

ElGamal PKCS

Suppose we have two users A and B . User B would like to send a message to user A . User A sets up his public information as follows. A first selects a large prime p_A , where $p_A - 1$ has a large prime factor (see [Pohl78]), and a generator g_A for $Z_{p_A}^*$. An integer x_A is chosen from $Z_{p_A}^*$ and kept private by A . A computes $y_A = g_A^{x_A} \bmod p_A$ and publicizes the triplet (g_A, y_A, p_A) .

For B to send a secret message to A , B first obtains (g_A, y_A, p_A) from the public directory. To encrypt a message $0 \leq K \leq p_A - 1$, B first selects a random $e \in Z_{p_A}^*$, computes

$$\begin{aligned} r_{A1} &= g^e \pmod{p_A} \\ r_{A2} &= K(y_A)^e \pmod{p_A} \end{aligned}$$

and sends (r_{A1}, r_{A2}) to A .

Upon receipt of (r_{A1}, r_{A2}) , A computes the following

$$\begin{aligned} M &= ((r_{A1})^{x_A})^{-1} r_{A2} \pmod{p_A} \\ &= ((g^e)^{x_A})^{-1} (y_A)^e K \pmod{p_A} \\ &= ((g^{x_A})^e)^{-1} (g^{x_A})^e K \pmod{p_A} \\ &= K \end{aligned}$$

Therefore, A is able to receive the intended message K . ElGamal was transformed into a PKCS from the Diffie-Hellman key agreement protocol (see Section 1.1). Like Diffie-Hellman, ElGamal is believed to be as difficult to attack as the Discrete Logarithm problem.

Using ElGamal to Broadcast

Extending the idea from Section 4.1.1 to broadcast ElGamal encrypted messages is straightforward. Whereas in section 4.1.1, we had only one message for each member of P , we now have two messages on which to apply the CRT. For user

u_B to broadcast a message K to a privileged set P of users, u_B will compute the ElGamal ciphertexts (r_{i1}, r_{i2}) for each $u_i \in P$, and solve the following equations

$$\begin{aligned} r_{1j} &= C_j \pmod{p_1} \\ r_{2j} &= C_j \pmod{p_2} \\ &\vdots \\ r_{tj} &= C_j \pmod{p_t} \end{aligned}$$

using the CRT for $j = 1, 2$. In other words, u_B creates C_1 and C_2 and broadcasts them to all users. To allow for users to discover whether they were intended recipients, for $u_k \notin P$, the equation $r_{kj} = 0$ is included above. Upon receipt of these two terms, $u_i \in P$ computes

$$\begin{aligned} r_{i1} &= C_1 \pmod{p_1} \\ r_{i2} &= C_2 \pmod{p_1} \end{aligned}$$

and recovers K via an ElGamal decryption. Users $u_k \notin P$ will recover $r_{k1} = r_{k2} = 0$.

4.1.3 The Sealed Lock

One problem with the secure lock model is that each user is required to possess an additional integer, to allow computation of the secure lock using the CRT. The notion of a *sealed lock* as presented in [Lin92], presents a different version of the implementations from the previous two sections. The sealed lock is an improvement for two reasons. They use ElGamal, as in [Chan92], but each user uses the same publically available prime p and corresponding generator g . Secondly, they use a lock other than the CRT. The lock is such that users are not required to each possess a separate, distinct integer.

Once again, user u_B will broadcast a message to all users, such that only a privileged subset P of users can recover the session key K from this message. First, individual encryptions of K are performed for each user in P . Second, a sealed lock C is constructed from these messages.

Suppose that a prime p and generator g for Z_p^* are publicly known to all users. Each user u_i chooses a random x_i and retains this as his private key. u_i computes $y_i = g^{x_i} \bmod p$ and publishes y_i as his public key. For u_B to broadcast K to the t members in P , $r_i = K(y_i)^K \bmod p$ is computed for each u_i and a sealed lock C is computed from them. Then C and $K_y = g^K \bmod p$ are broadcast to all users.

The sealed lock C is based on the following Theorem from [Lin92].

Theorem 4.1 *Let $r_1, r_2, \dots, r_n$ be n elements from Z_p^* . There exists an integer C , called a sealed lock, such that*

$$r_i \equiv \left\lfloor \frac{C}{(p+1)^{i-1}} \right\rfloor \pmod{p+1}, i = 1, 2, \dots, n.$$

The integer C satisfying this equation is

$$C = \sum_{i=1}^n r_i (p+1)^{i-1} \quad (4.2)$$

Therefore, (4.2) is used to create the sealed lock C while Theorem 4.1 is used by user u_i to recover his r_i . To recover K , u_i computes $K = K_y^{-x_i} r_i$. To solve the user membership question, this scheme simply assigns $r_i = 0$ for those users not in P .

The following requirements for the lock are stated in [Lin92].

1. C should be opened only by users in P .
2. C should be dependent upon the session key K .

The first point is somewhat misleading. Once again the index i required to compute the equation from Theorem 4.1 is publicly known. Therefore, any user can open C , but only to reveal selected r_i 's. Hence the security of the scheme will once again rely on the security of the underlying cryptosystem (which will be discussed in upcoming paragraphs). The second point merely ensures that C will differ based on the random session key, i.e., the same C will not be constructed next time the same members of P form the privileged set. This is true since a different K is used each time, and the r_i depend upon K .

The following properties of this broadcast system are stated in [Lin92].

1. Only a single copy of a message from a sender has to be encrypted and transmitted.
2. For a group of users, only 1 common secret key is required.
3. The key can be changed without changing user's keys.

The first two points are very similar to those from section 4.1.1. The first point is misleading if not incorrect. It is two ciphertexts that are transmitted; C and K_y . The computation of K_y is a single encryption used to allow the recovery of K by members in P . However, there are also t other encryptions performed by the sender, i.e., computation of the r_i 's.

The second point merely states the ability of the system to provide users in P with a common key. The third point is based on the assumption that the underlying cryptosystem is secure. Even if the session key is compromised, the secret keys of each of the users need not be changed. Of course, this implementation inherits the problem of updating the session key (Section 4.1.1) in that multiple encryptions for each user must be performed, before a new broadcast message can be created.

4.2 Parallel Broadcasting

In this section we present a conceptual framework for broadcasting a session key K with user anonymity. We introduce a foundation for the parallel construction of a broadcast message, such that only members of the privileged set can recover K from this message. Users have no knowledge of the other members of P , nor do they require this knowledge for recovery of K .

Once the privileged set P has been defined, T generates a broadcast encryption key K_P using the secret keys of the users in P as follows:

$$K_P = f(s_1, s_2, \dots, s_t), \quad (4.3)$$

for an appropriate function f . T then computes

$$B = E_{K_P}(K), \quad (4.4)$$

Figure 4.2: Parallel Encryption of a session key K

where B is the broadcast message. E is an encryption function parameterized by the key K_P . This process is illustrated in Figure 4.2. f and E are designed such that only knowledge of a single s_i from $u_i \in P$ is required to recover K from B , i.e.,

$$K = D_{s_i}(B), \forall u_i \in P,$$

for the decryption function D corresponding to E . This will become more meaningful with the schemes presented in Section 4.4 and Section 4.5.

An important difference from previous methods is the ease with which one can update K . By this general approach, one need only recalculate and broadcast (4.4) to establish a new session key K , whereas those from Section 4.1 require separately re-encrypting the new K for each user, followed by recalculation of the secure/sealed lock. Contrast the construction of the broadcast message in Figure 4.1 with Figure 4.2. In fact, this model could more suitably be called a secure/sealed lock, with master key K_P and equally effective keys s_i for each user $u_i \in P$.

4.3 A Lower Bound

In this section, we present a lower bound for the size of the broadcast message for a certain class of parallel broadcast schemes. First, we recall some standard

information theoretic properties. For further reference, consult [Jum290, Shan49].

Given a finite set X of size n and a probability distribution $\{p(x_i)\}_{x_i \in X}$, the *entropy of X* is defined to be

$$H(X) = \sum_{i=1}^n p(x_i) \log_b \left(\frac{1}{p(x_i)} \right)$$

where $\log_b(\frac{1}{0})$ is defined to be 0. The subscript b refers to our base of reference. For our purposes, we will be using $b = 2$ and hence computing *bits* of entropy.

Entropy has some useful interpretations. $H(X)$ defines one's average uncertainty in X , i.e., uncertainty about which element of X has been chosen, given the probability distribution. $H(X)$ is also useful for approximating the minimum number of bits required to encode elements of X .

The range of $H(X)$ is $0 \leq H(X) \leq \log |X|$, where the lower bound is obtained when $p(x_i) = 1$ for some i , while $p(x_j) = 0, \forall j \neq i$. The upper bound is achieved when $p(x_i) = \frac{1}{|X|}, \forall i$.

The *equivocation of X given Y* is defined to be

$$H(X|Y) = \sum_{i=1}^n \sum_{j=1}^m p(y_j) p(x_i|y_j) \log \left(\frac{1}{p(x_i|y_j)} \right)$$

where Y is a finite set of size m . $H(X|Y)$ can be thought of as the uncertainty in X , given that Y has been observed *a priori*. Note that $H(X|Y) \geq 0$. Given $t + 1$ sets $X_1, X_2, \dots, X_t, Y$,

$$H(X_1 \dots X_t | Y) = H(X_1 | Y) + H(X_2 | X_1 Y) + \dots + H(X_t | X_1 \dots X_{t-1} Y)$$

The *mutual information* or *transinformation* of X and Y is defined as

$$I(X|Y) = H(X) - H(X|Y)$$

and can be thought of as the amount of information that Y reveals about X . Note that $I(X, Y)$ and $I(X; Y)$ are also equivalent notations. If X and Y are independent, then $H(X|Y) = H(X)$ and $I(X|Y) = 0$. In other words Y contributes no information about X . Similarly,

$$I(X|YZ) = H(X) - H(X|YZ),$$

for finite sets X, Y and Z . Transinformation has the properties that

$$I(X|Y) = I(Y|X),$$

$$I(X|Y) \geq 0.$$

From the latter statement, note that $H(X) \geq H(X|Y)$.

The *conditional transinformation* of the pair X, Y given Z is defined as

$$I_Z(X|Y) = H(X|Z) - H(X|YZ),$$

and can be interpreted as the amount of information that Y provides about X , given that Z has already been observed. From this we obtain

$$I(X|YZ) = I(X|Y) + I_Y(X|Z).$$

4.3.1 Properties of Model

In this section, we take the general model for broadcasting session keys as given in Section 4.2, and add specific conditions to define a formal model, hereafter referred to as the model for Broadcasting with Information Theoretic Security (BITS). We then prove a lower bound on the size of the broadcast message required to broadcast a session key within this model.

Let $\mathcal{B}$ and K be the respective broadcast message and session key from Section 4.2 and s_i be the share of $u_i \in P$. Without loss of generality, let $\mathcal{S}_P = (s_1, \dots, s_t)$ be the t shares of the users in P . The most important result for this model is Theorem 4.2. The following are assumptions that are necessary for obtaining the result of the theorem (via direct use in the proof of the theorem, and in the lemmas that are essential to proving the theorem).

A1 $H(s_i|\mathcal{B}) = H(s_i)$. The broadcast message does not decrease the uncertainty in a user's share.

A2 $H(K|\mathcal{B}) = H(K)$. The broadcast message does not decrease the uncertainty in the session key.

A3 $H(s_i | s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_t) = H(s_i)$. The share of one user is independent of the shares of other users.

A4 $H(K | \mathcal{B} s_i) = 0$. The session key is uniquely defined by the broadcast message and the share of any member of the privileged set.

A5 $H(K | \mathcal{S}_P) = H(K)$. The shares of the users alone, reveal no information about the session key.

A6 $H(\mathcal{B} | \mathcal{S}_P K) = 0$. The shares of the users in P , and the session key, define the broadcast message.

The following are two technical lemmas required for the proof of Theorem 4.2. Lemma 4.1 merely enforces the fact that the broadcast message $\mathcal{B}$ only contains information relating to the shares of the users in P and the session key K .

Lemma 4.1 *For the BITS model, $H(\mathcal{B}) = I(\mathcal{B} | \mathcal{S}_P K)$.*

Proof:

$$\begin{aligned}
 I(\mathcal{B} | \mathcal{S}_P K) &= I(\mathcal{B} | s_1) + I_{s_1}(\mathcal{B} | s_2) + I_{s_1 s_2}(\mathcal{B} | s_3) + \dots + I_{\mathcal{S}_P}(\mathcal{B} | K) \\
 &= H(\mathcal{B}) - H(\mathcal{B} | s_1) + H(\mathcal{B} | s_1) - H(\mathcal{B} | s_1 s_2) \\
 &\quad + H(\mathcal{B} | s_1 s_2) - H(\mathcal{B} | s_1 s_2 s_3) + \dots + H(\mathcal{B} | \mathcal{S}_P) \\
 &\quad - \underbrace{H(\mathcal{B} | \mathcal{S}_P K)}_{0 \text{ by A6}} \\
 &= H(\mathcal{B})
 \end{aligned}$$

□

Lemma 4.2 *Let $\mathcal{D} \subset P$ be a non-empty subset of privileged participants such that $|\mathcal{D}| \leq (t-1)$, and let $\mathcal{S}_{\mathcal{D}}$ be the set of shares of participants in $\mathcal{D}$. Also let s_i be the share of $u_i \in P$ such that $u_i \notin \mathcal{D}$. Given a session key K and broadcast message $\mathcal{B}$ for the BITS model, $H(s_i) - H(s_i | \mathcal{B} \mathcal{S}_{\mathcal{D}}) \geq H(K)$.*

Proof:

The term $H(s_i K | \mathcal{BS}_D)$ simplifies to $H(s_i | \mathcal{BS}_D) + H(K | \mathcal{BS}_D s_i)$ or $H(K | \mathcal{BS}_D) + H(s_i | \mathcal{BK} \mathcal{S}_D)$ (cf. [Capo92, Lemma 3.3]). Therefore, we can write

$$\begin{aligned} H(s_i | \mathcal{BS}_D) + \overbrace{H(K | \mathcal{BS}_D s_i)}^{0 \text{ by A4}} &= \overbrace{H(K | \mathcal{BS}_D)}^{0 \text{ by A4}} + H(s_i | \mathcal{BK} \mathcal{S}_D) \\ H(s_i | \mathcal{BS}_D) &= H(s_i | \mathcal{BK} \mathcal{S}_D) \end{aligned} \quad (4.5)$$

We also have,

$$\begin{aligned} I_{\mathcal{B}}(s_i | K) &= I_{\mathcal{B}}(K | s_i) \\ \overbrace{H(s_i | \mathcal{B})}^{H(s_i) \text{ by A1}} - H(s_i | \mathcal{BK}) &= \overbrace{H(K | \mathcal{B})}^{H(K) \text{ by A2}} - \overbrace{H(K | \mathcal{B} s_i)}^{0 \text{ by A4}} \\ H(s_i) - H(s_i | \mathcal{BK}) &= H(K) \end{aligned} \quad (4.6)$$

And,

$$\begin{aligned} I_{\mathcal{B},K}(s_i | \mathcal{S}_D) &= H(s_i | \mathcal{BK}) - H(s_i | \mathcal{BK} \mathcal{S}_D) \geq 0 \\ \Rightarrow H(s_i | \mathcal{BK}) &\geq H(s_i | \mathcal{BK} \mathcal{S}_D) = H(s_i | \mathcal{BS}_D). \end{aligned} \quad (4.7)$$

The last equality in (4.7) is obtained from (4.5). The result follows by applying (4.7) to (4.6). $\square$

On its own, Lemma 4.2 implies that if the entropy (uncertainty) of a privileged user's share s_i is equal to the entropy of the session key K , then given the broadcast message, any privileged user(s) have no uncertainty in s_i , as illustrated by the following corollary.

Corollary 4.1 *For the BITS model, if $H(s_i) = H(K)$ then $H(s_i | \mathcal{BS}_D) = 0$.*

Proof:

Follows from lemma 4.2, where $H(s_i | \mathcal{BS}_D) \leq H(s_i) - H(K)$. $\square$

In a perfect secret sharing scheme, the size of the shares must be at least as large as the secret key (see Section 3.1). The following corollary highlights the fact that the number of bits used to encode a user's secret must be at least as large as those used to encode the session key.

Corollary 4.2 *For the BITS model, $H(s_i) \geq H(K)$.*

Proof:

From lemma 4.2, we have

$$H(s_i) \geq H(K) + H(s_i | \mathcal{BS}_D).$$

Since $H(s_i | \mathcal{BS}_D) \geq 0$, the result follows. $\square$

The following theorem will show that for the BITS model, the minimum number of bits required to encode the broadcast message $\mathcal{B}$ is at least as large as t copies of the session key K , where $|P| = t$.

Theorem 4.2 *For the BITS model, $H(\mathcal{B}) \geq tH(K)$.*

Proof:

Recall that $\mathcal{S}_P$ is the set of t shares of users in P .

$$\begin{aligned} H(\mathcal{B}) &= I(\mathcal{B} | \mathcal{S}_P K) && \text{(by Lemma 4.1)} \\ &= I(\mathcal{B} | \mathcal{S}_P) + I_{\mathcal{S}_P}(\mathcal{B} | K) \\ &= I(\mathcal{S}_P | \mathcal{B}) + I_{\mathcal{S}_P}(K | \mathcal{B}) \\ &= H(\mathcal{S}_P) - H(\mathcal{S}_P | \mathcal{B}) + \overbrace{H(K | \mathcal{S}_P)}^{H(K) \text{ by A5}} - \overbrace{H(K | \mathcal{BS}_P)}^{0 \text{ by A4}} \end{aligned} \quad (4.8)$$

$$\text{Now,} \quad H(\mathcal{S}_P) = H(s_1) + H(s_2) + \cdots + H(s_t) \quad (\text{by A3})$$

$$\begin{aligned} H(\mathcal{S}_P | \mathcal{B}) &= H(s_1 s_2 \dots s_t | \mathcal{B}) \\ &= H(s_1 | \mathcal{B}) + \sum_{j=2}^t H(s_j | \mathcal{B} s_1 \dots s_{j-1}) \end{aligned}$$

So, completing (4.8), we have

$$\begin{aligned} H(\mathcal{B}) &= \overbrace{H(s_1) - H(s_1 | \mathcal{B})}^{0 \text{ by A1}} + \sum_{j=2}^t (H(s_j) - H(s_j | \mathcal{B} s_1 \dots s_{j-1})) \\ &\quad + H(K) \\ &\geq (t-1)H(K) + H(K) && (\text{by Lemma 4.2}) \\ &= tH(K) \end{aligned}$$

□

Therefore, the size of the broadcast message for a scheme satisfying the properties of Section 4.3.1 is equivalent to sending individual encryptions except for the fact that no addressing is required. Therefore, the additional $\lg t$ bits required for sending messages individually, is not necessary here.

4.4 Parallel Broadcasting Using Secret Sharing

Recall that a (t, n) *threshold scheme* implies a method for sharing a secret key K among a finite set U of n users such that a subset P of at least $t \leq n$ users from U can recover K while no group of size less than t can do so (see Chapter 3). A trusted center T randomly selects a key K and uses a concealer function (see Figure 3.1) to produce n shares or shadows on input K . The n shares are then securely distributed to the n users, each user receiving exactly one share. To recover K , the users in the privileged set P , where $|P| \geq t$, combine their shares and input them to a revealer function (see Figure 3.2) to produce K . For every set X , where $|X| < t$, if inputting less than t shares to the revealer function will give the users in X no advantage in obtaining K , the threshold scheme is said to be *perfect*.

The major problem with using secret sharing to directly broadcast session keys is that it requires the shares of all members of the set P to be input to the revealer function. In reality, this requires either the physical presence of each user or some way of securely communicating their shares. We proceed to alter the method slightly to allow the broadcast of session keys, without either of the preceding requirements.

Let each user u_i continue to share a long-term secret s_i with T . T computes a random session key K and will share it with a privileged set P where $|P| = t$. T proceeds to compute the following:

$$B = g(K, s_1, s_2, \dots, s_t) \quad (4.9)$$

for an appropriate “share-generating function” g , where s_i is the secret share of $u_i \in P$. B is a set of t shares distinct from the shares input to the function g , and

is broadcast to all users. Just as in Section 4.2, each user $u_i \in P$ needs only their share s_i to recover K .

Notice the relation that (4.3) and (4.4) of Section 4.2 have with (4.9). If f from (4.3) simply returns its parameters as output, then E from (4.4) and g from (4.9) are equivalent. This suggests the use of secret sharing to essentially achieve encryption. (In fact, this is done by Gong in [Gong94].)

Notice that we are broadcasting t shares which were formed from the session key K and the shares of each of the members of P . The intent is that a perfect $(t + 1, n)$ threshold scheme for sharing the secret K is implied by g , where t shares from B plus any one share s_i for $u_i \in P$ reaches the threshold. Recall that in a perfect threshold scheme, $(t + 1) - 1$ shares (i.e., the broadcast message B in this case) reveal no information about the intended secret.

This general idea was considered by Lai *et al.* [Lai89], and subsequently by Berkovits [Berk92]. In the following sections, we present both schemes from [Berk92]. One satisfies the properties of Section 4.3.1, allowing us to improve it to reach the lower bound of Theorem 4.2.

In both schemes, random shares can be added to ensure that the same set P does not always have the same encryption key K_P from (4.3) for different sessions. For simplicity, we do not include this into our description of the schemes.

4.4.1 Using Shamir Interpolation to Broadcast

Berkovits [Berk92] uses the threshold scheme from Shamir [Sham79] to allow a trusted center T to broadcast messages to a privileged set P as follows.¹ Each user $u_i \in U$ shares a secret point $(x_i, y_i) \in Z_p \times Z_p$ with T , where $x_i \neq x_j$ for $i \neq j$. To share a session key with members of P , T first selects a random $K \in Z_p^*$. T then inputs t points $(x_i, y_i), \forall i : u_i \in P$ and $(0, K)$ to g in (4.9). The function g will first produce a polynomial of degree t ,

$$p(x) = K + a_1x^1 + \cdots + a_tx^t \pmod{p}. \quad (4.10)$$

¹We simplify the scheme somewhat for our purposes.

Note that $t + 1$ points are required to uniquely define $p(x)$. $\mathcal{B}$ from (4.9) will be made up of t points in $Z_p^* \times Z_p^*$ on the polynomial, distinct from any shares of a user $u_i \in P$. Since t distinct points are both input to and output from g and $t \leq n$, we require $p \geq 2n + 1$. The reconstruction of $p(x)$ by each user is relatively expensive. Given t points, $O(t^2)$ multiplications in Z_p^* are required [Denn83, page 53]. From $\mathcal{B}$ and his own share, each $u_i \in P$ has enough points to reconstruct $p(x)$, and compute $p(0) = K$. A user in possession of only $\mathcal{B}$ does not have enough points to reconstruct $p(x)$ and hence has no advantage in recovering K .

For this scheme, the size of $\mathcal{B}$ is $2t \log_2 p$ bits, and $H(s_i) = 2 \log_2 p$ for a randomly chosen point (x_i, y_i) on $p(x)$, whereas $H(K) = \log_2 p$. We do not achieve the lower bound of Theorem 4.2 for two reasons. The first is that $H(\mathcal{B}) = 2tH(K)$. The second is that assumption A3 from Section 4.3.1 is not satisfied, i.e., the shares of the users are not independent of one another. The following section solves both of these problems.

Meeting the Lower Bound

To improve on the above scheme, we employ an idea of Krawczyk [Kraw93] (used in another context). Observe that after constructing $p(x)$, user $u_i \in P$ recovers one of the $t + 1$ polynomial coefficients, namely the constant, as the session key K .

Define the shares created by $\mathcal{B}$ to be the coefficients $a_1, \dots, a_t$ of $p(x)$. To recover $p(x)$, $u_i \in P$ will create the polynomial in (4.10) missing only K (notice that users will possess all the coefficients of the polynomial, except for the constant K). Using his secret share (x_i, y_i) (a point on $p(x)$), u_i can easily solve for the constant of the polynomial, K . This can be done with a simple substitution, followed by an application of Horner's rule with $O(t)$ multiplications. Note the size of $\mathcal{B}$ is now $t \log_2 p$ bits where the session key K is $\log_2 p$ bits.

We note that this scheme both satisfies the conditions for the model from Section 4.3.1 and realizes the lower bound of Theorem 4.2, and is thus optimal in this regard. Notice that we satisfy the independence assumption A3 from Section 4.3.1 for this scheme by defining the share of each user to simply be the y -component. In

this way, the shares of each user are independent of one another. However, this is done only to comply with the assumptions necessary to arrive at the result of Theorem 4.2. As explained below, this alteration is made at the expense of decreasing the security of the scheme. Section 4.4.3 examines this property further.

Since only the secrecy of the y -component is maintained, we would now have $H(s_i) = \log_2 p$ for a randomly chosen point (x_i, y_i) on $p(x)$. Given the broadcast message $\mathcal{B}$ and the point (x_j, y_j) on $p(x)$ for user $u_j \in P$, we have $H(s_i | \mathcal{B} \mathcal{S}_{\mathcal{D}}) = 0$, where $\mathcal{S}_{\mathcal{D}}$ need only consist of s_j (see Lemma 4.2). This results from the fact that user u_j has the ability to recover $p(x)$ and the corresponding y -component of user u_i 's share, since the secrecy of the x -component is not maintained. Note also that the implication of having $H(s_i | \mathcal{B} \mathcal{S}_{\mathcal{D}}) = 0$ does not necessarily compromise the security of s_i in practice (even though it does in this case), i.e. it may be computationally infeasible to find s_i . This point is discussed further in Section 4.4.3.

4.4.2 Using Vectors to Broadcast

Berkovits [Berk92] presented an additional scheme, related to the secret sharing scheme of Brickell [Bric89]. The work done by each user is reduced from interpolation to a number of multiplications and additions. Like the original scheme of Section 4.4.1, an inability to satisfy A3 from Section 4.3.1 prevents this scheme from being compared to the model of Section 4.3.1.

Each user u_i shares a triplet $(\vec{v}_i, y_i, z_i)$ with T and knowledge of a prime p chosen by T .² The latter two components of $(\vec{v}_i, y_i, z_i)$ are integers, while the first is a vector of m integers, each chosen from Z_p . T creates the broadcast message as follows.

- Solve for the vector $\vec{W}$ such that $\vec{W} \cdot \vec{v}_i = y_i \bmod p$ for users in the privileged set P , while $\vec{W} \cdot \vec{v}_i \neq y_i \bmod p$, for users not in P .
- Solve for the vector $\vec{y}$ such that $\vec{y} \cdot \vec{v}_i = z_i^{-1} \bmod p$.
- Broadcast the vector $\vec{C}$ where $\vec{C} = \vec{W} + K\vec{y} \bmod p$ for the session key K .

²The specification of a prime p was not considered in [Berk92]. We include it to maintain a consistency with the previous scheme.

User u_i recovers K from $\tilde{C}$ by performing the following.

$$\begin{aligned}
 \tilde{C} &= \tilde{W} + K\tilde{y} \pmod{p} \\
 \tilde{C} - K\tilde{y} &= \tilde{W} \pmod{p} \\
 (\tilde{C} - K\tilde{y}) \cdot \tilde{v}_i &= y_i \pmod{p} \\
 (\tilde{C} \cdot \tilde{v}_i) - (K\tilde{y} \cdot \tilde{v}_i) &= y_i \pmod{p} \\
 K(\tilde{y} \cdot \tilde{v}_i) &= (\tilde{C} \cdot \tilde{v}_i) - y_i \pmod{p} \\
 K &= ((\tilde{C} \cdot \tilde{v}_i) - y_i)(\tilde{y} \cdot \tilde{v}_i)^{-1} \pmod{p} \\
 K &= ((\tilde{C} \cdot \tilde{v}_i) - y_i)z_i \pmod{p}
 \end{aligned}$$

4.4.3 Preventing Insider Attacks

In the conversion of a secret sharing scheme to a broadcasting scheme, we inherit some problems that we have previously encountered. More specifically, the publication of the x -component of a user's share (as discussed in Section 4.4.1), results in the compromise of a user's y -component.

This problem was addressed for threshold schemes in Section 3.4. The same techniques that were used to handle "disenrollments" can be applied here. However, they remain impractical as the size of a user's share becomes unacceptably large.

A more practical solution is to make use of one-way hash functions. This was done by Gong [Gong94]. The idea is similar to those presented for threshold schemes in Section 3.4 by Zheng *et al.* [Zhen94].

Suppose that we have a one-way, keyed hash function h (h can simply be thought of as an encryption function that compresses its input). Rather than using each user's *actual* share (x_i, y_i) as input to the concealer function, T uses $(h(x_i, r), h(y_i, r))$ where $r \in \mathbb{Z}_p^*$ is a random integer selected by T that will be broadcast to all users.

If the x_i of u_i is public, then some u_j can still recover $h(x_i, r)$. Given that $u_j \in P$ has also recovered K (and hence the polynomial), u_j can also recover the corresponding y -coordinate $h(y_i, r)$. However, knowledge of $(h(x_i, r), h(y_i, r))$ only allows u_j to recover the current session key K ; something that he has recovered on

his own already. Note that for the recovery of the next key K' , $(h(x_i, s), h(y_i, s))$ will be used for the random broadcast message $s \in Z_p^*$. The one-way property of h precludes u_j from recovering y_i .

4.5 Parallel Broadcasting Using Number Theory

The following sections discuss two implementations of the model presented in section 4.2. The security of each relies on certain number theoretic assumptions. Thus, they are computationally secure as opposed to unconditionally secure. Therefore, the model of Section 4.3 cannot be applied to these problems and the lower bound of Theorem 4.2 does not apply.

The first implementation is based on the assumption that it is difficult to determine the quadratic residuosity of an integer. The second is based on the assumption that it is difficult to compute t^{th} roots of an integer, modulo N , where the factorization of N is not known.

Before we present the scheme based on quadratic residuosity, we motivate the setup with the following preliminary scheme.

Let T share a distinct, independent prime number p_i with each of n users. The broadcast key K_P from (4.3) is simply N where $N = p_1 \cdots p_t$. T randomly selects an integral session key K in the range $0 < K < \min\{p_1, p_2, \dots, p_t\}$. The broadcast message C from (4.4) is the integer $C = K^{-1} \bmod N$. Upon receipt of C , each $u_i \in P$ computes

$$K = C^{-1} \pmod{p_i} \quad (4.11)$$

In accordance with the model of Section 4.2, given only the single piece that each $u_i \in P$ shares with T , allows recovery of K . Consider that if $KK^{-1} \equiv 1 \pmod{N}$, then $KK^{-1} \equiv 1 \pmod{p_i}$ for each prime factor p_i of N . Therefore, for each p_i that $K < p_i$ holds true, computation of K from K^{-1} with only p_i is realized. This is done by reducing K^{-1} modulo p_i , and computing its inverse in $Z_{p_i}^*$.

If the each of the p_i are encoded with $\log_2 p$ bits, for $p = \max(p_i)$, then the size of the broadcast message C is $t \log_2 p$ bits. This happens to satisfy the lower bound on

the size of the broadcast message of Section 4.3 despite the fact that the bound (as it is found in Section 4.3) cannot be applied to the current section. This is not all too surprising though since a bound of having to send t times as much information to broadcast a message to t users sounds intuitively correct.

The amount of work performed by the center to produce C is at most $2t \lceil \log_2(p) \rceil$ modular multiplications in Z_N^* . The calculation of C can be done using Euclid's Extended Algorithm (see Section A.1). Since each user i requires only their prime p_i to recover K (in the manner described above), at most $2 \lceil \log_2(p_i) \rceil$ modular multiplications in $Z_{p_i}^*$ are required once C has been reduced modulo p_i .

Thus, we have constructed a scheme whereby T can encrypt a message K using the secrets of *all* users in P , yet only a *single* key from a user in P is sufficient to recover K . However, this scheme is not very secure since coalitions of users can, over time, recover information about the secret keys of other users. This is a result of the fact that the equation $KK^{-1} \equiv 1 \pmod{p_i}$ is valid for all $u_i \in P$. By the construction of K^{-1} , the term $KK^{-1} - 1$ is divisible by all of the primes that divide N . Hence each time u_i recovers a session key, he learns additional information about the primes of all other users in the privileged set.

4.5.1 Using Quadratic Residues to Broadcast

In this section, we present a scheme for broadcasting a session key K which satisfies the model of Section 4.2. The scheme is based on some properties of quadratic residues. We begin by introducing quadratic residues and their relevant properties.

Quadratic Residuosity

Given a prime number p , q is a *quadratic residue mod p* (denoted $q \in QR_p$) if $\exists x \in Z_p^*$ such that $x^2 \equiv q \pmod{p}$ for $q \in Z_p^*$. If q is not a quadratic residue, then q is a *quadratic non-residue* (denoted $q \in QNR_p$). For example, $2 \in QR_7$ since $3^2 \equiv 2 \pmod{7}$, while $3 \in QNR_7$ since $\nexists x \in Z_7^*$ such that $x^2 \equiv 3 \pmod{7}$.

Now given an integer $N = p_1 p_2 \cdots p_t$, where each p_i is a distinct prime,

$$q \in QR_N \Leftrightarrow q \in QR_{p_1} \cap QR_{p_2} \cap \cdots \cap QR_{p_t} \quad (4.12)$$

whereas

$$q \in QNR_N \Leftrightarrow q \in QNR_{p_1} \cup QNR_{p_2} \cup \cdots \cup QNR_{p_t} \quad (4.13)$$

Notice that in (4.12), all of $q \in QR_{p_i}$ must be true, while in (4.13), only one of $q \in QNR_{p_i}$ need be true.

Given an integer q , one can determine whether or not q is a quadratic residue modulo a prime p , by performing the following test ([Denn83, pp. 113–114]). Calculate $q^{\frac{p-1}{2}}$ and perform the following comparison:

$$\begin{aligned} q^{\frac{p-1}{2}} &\equiv 1 \pmod{p} \Rightarrow q \in QR_p \\ &\equiv -1 \pmod{p} \Rightarrow q \in QNR_p \end{aligned} \quad (4.14)$$

The calculation $q^{\frac{p-1}{2}}$ is denoted by $\left(\frac{q}{p}\right)$ and is known as the *Legendre* symbol (for a prime p). For a composite N , the term $\left(\frac{q}{N}\right)$ is known as the *Jacobi* symbol and can be computed as

$$\left(\frac{q}{N}\right) = \left(\frac{q}{p_1}\right) \cdots \left(\frac{q}{p_t}\right) \quad (4.15)$$

where $N = p_1 p_2 \cdots p_t$. From (4.14), $\left(\frac{q}{N}\right)$ will either be 1 or -1 . Notice from (4.13) and (4.14) that if $\left(\frac{q}{N}\right) = -1$, at least one of $\left(\frac{q}{p_i}\right) = -1$ and thus $q \in QNR_N$. However, having $\left(\frac{q}{N}\right) = 1$, does not imply that $q \in QR_N$. To illustrate this fact, suppose that $\left(\frac{q}{p_i}\right) = -1$ and $\left(\frac{q}{p_{i+1}}\right) = -1$ while $q \in QR_{p_j}$ for $j \neq i$ and $j \neq i+1$. Therefore, from (4.12), $q \in QNR_N$, but calculation of (4.15) will reveal $\left(\frac{q}{N}\right) = 1$.

The computation of $\left(\frac{q}{p}\right)$ requires at most $2\lceil \lg((p-1)/2) \rceil$ modular multiplications in Z_p^* [Ries85]. For a composite N , the product of t primes, no more than t times as many multiplications are required, given that p represents the largest prime. Therefore, given the factorization of n , computation of $\left(\frac{q}{N}\right)$ is proportional to the equivalent computation with a prime. In other words, both computations are quite simple.

Despite the fact that the above computation of $\left(\frac{q}{N}\right)$ is defined in terms of the factorization of N , there are simple methods of computing $\left(\frac{q}{N}\right)$ without knowing

N 's factorization. This is an important factor for the construction of the quadratic residuosity broadcast scheme.

Another important property to consider about quadratic residues is their cardinality. Consider that Z_p^* has $p - 1$ elements. Exactly $\frac{1}{2}$ of these are quadratic residues, while another $\frac{1}{2}$ are quadratic non-residues. This is a result of the fact that $x^2 \equiv (-x)^2 \pmod{p}$, i.e. $f(x) = x^2 \pmod{p}$ is a 2-to-1 function on Z_p^* . For a composite $N = p_1 p_2 \cdots p_t$, we know from (4.12) that $q \in QR_N$ iff $q \in QR_{p_i}$ for all $p_i | N$. Therefore 2^{-t} of the elements in Z_N^* are quadratic residues.

Using Quadratic Residues to Broadcast

We now present an implementation based on quadratic residuosity, that fits the framework of section 4.2.

T shares a distinct, random prime number p_i with each of n users. Let $K = k_0 k_1 \cdots k_{l-1}$ be the binary representation of the session key. K_P is constructed as follows. First T calculates

$$N = p_1 p_2 \cdots p_t \quad (4.16)$$

where p_i is the secret prime that T shares with user $u_i \in P$. T will produce a y such that $y \in QNR_N$. We require that $y \in QNR_{p_i}$ for each p_i in (4.16), which may be done by choosing random $y_i \in Z_{p_i}^*$ and using (4.14) to determine if $y_i \in QNR_{p_i}$. Alternatively, each u_i could share a pre-defined y_i with T . In either case, T can then use the Chinese Remainder Theorem (see Appendix A.2) with each y_i and p_i to solve for y such that $y \in QNR_N$. Here, $K_P = (y, N)$ serves as the broadcast key.

For the encryption function E in (4.4), we use a variant of the method of *probabilistic encryption* from [Gold84]. Sending one bit of K will require broadcasting a $\log_2 N$ bit integer. For T to broadcast a bit k_b such that only $u_i \in P$ can recover k_b , T selects a random $x \in Z_N^*$ and computes $C_b = x^2 y^{k_b} \pmod{N}$ and broadcasts C_b to all users. Note that $k_b \in [0, 1]$ while $C_b \in [1..N - 1]$.

Before continuing, we first point out that the product of a quadratic residue and a quadratic non-residue is a quadratic non-residue. Consider our current situation

and assume the opposite is true, i.e., $x^2y \equiv z^2(\text{mod } N)$ for some $z \in Z_N^*$. However, this would imply that $y \equiv (zx^{-1})^2(\text{mod } N)$, which cannot be true, since $y \in QNR_N$.

Now consider the following two possibilities. If $k_b = 0$ then $C_b \equiv x^2 \pmod{N}$ and thus $C_b \in QR_N$. From (4.12), this implies $C_b \in QR_{p_i}$ for each prime p_i . By (4.14), each u_i who possesses a prime divisor p_i of N can determine that $C_b \in QR_N$ and conclude that $k_b = 0$. If $k_b = 1$, then $C_b \equiv x^2y \pmod{N}$ and thus $C_b \in QNR_N$. Since y was chosen such that $y \in QNR_{p_i}$ for each p_i in the factorization of N , again by (4.14) each privileged user can determine that $C_b \in QNR_N$ and conclude that $k_b = 1$. Users $u_j \notin P$ cannot recover k_b as they lack the appropriate primes p_i .

The size of the broadcast message is $l \log_2 N$ bits, where N is encoded with $t \log_2 p$ bits, for $p = \max(p_i)$ and an l -bit session key K . This exceeds the lower bound on the size of the broadcast message of Section 4.3 by a factor of l . This clearly is not a practical method of broadcasting. However, it does illustrate a secure scheme, built upon the framework presented in section 4.2.

Once a suitable y has been selected, the amount of work performed by the center to produce C_b is at most 2 modular multiplications in Z_N^* . To broadcast all of K , this process is repeated l times. Each user i using only their prime p_i to recover K (in the manner described above), requires at most $2\lceil \log_2(p_i) \rceil$ modular multiplications in $Z_{p_i}^*$ for each bit of C received (subsequent to the reduction of the $\log_2 N$ bit C with the modulus p_i). Since l bits are broadcast, this operation is repeated l times. Due to the significant degree of data expansion, we consider the scheme to be of theoretical interest only.

The security of this scheme is based on the assumption that an opponent can not determine the quadratic residuosity of an integer $q \text{ mod } N$ without knowledge of N 's prime factors. Given an integer $q \in Z_N^*$ and N , it is shown in [Gold84] that if determining quadratic residuosity was easy to solve for some q , then it could be solved easily for all q . Yet, this was for an N , the product of just 2 primes.

The importance of the fact that the Jacobi symbol $\left(\frac{q}{N}\right)$ can be computed efficiently without knowledge of the factorization of N is evident in the following.

Suppose an eavesdropper X were to know N for a certain broadcast of a session key K . Further suppose that there are an odd number of users in P , i.e., $t = 2k + 1$ for some positive integer k . Given that bit j of the session key is 1, then $C_j = x^2y \pmod{N}$ for some random $x \in Z_N^*$. X is then able to compute $\left(\frac{C_j}{N}\right) = -1$ and conclude the value of the j^{th} bit of the session key.

Yet this problem can be easily overcome by ensuring that N is the product of an even number of primes. This can be done by simply including a random prime in N , when there are an odd number of users in P . This allows us to achieve the same security as in [Gold84].

4.5.2 Using Exponentiation to Broadcast

In this section, we present a method of broadcasting a session key K , using the symmetric encryption function presented by Pohlig and Hellman in [Pohl78]. The scheme is similar to RSA (see Section 1.1) except that users who intend to communicate share a prime p . In the two party version, one user would have the exponent e while the other would have the exponent d (though from their information alone, each could compute the other's key). Encryption and decryption proceed as in RSA. Each key e and d can be used for either encryption or decryption. The scheme is "odd" given that it is a two-key symmetric system, where the term two-key is usually reserved for asymmetric ciphers, i.e., RSA.

Once again, T securely shares a distinct, random prime p_i with each of the n users. T will also compute an $e \in Z_{p-1}^*$ and maintain its secrecy. Each user is also given an integer $d_i \in Z_{p-1}^*$ such that $ed_i \equiv 1 \pmod{p-1}$. T is not required to keep the d_i . T will compute N as in (4.16) and calculate

$$C = K^e \pmod{N} \quad (4.17)$$

as the encryption of the session key K . C represents the broadcast message. Upon receipt of C , $u_i \in P$ computes

$$K \equiv (C \bmod p_i)^{d_i} \pmod{p_i}$$

$$\equiv K^{ed_i} \pmod{p_i}$$

This scheme is similar to the one that was used to introduce Section 4.5. In that scheme, the inverse of the key was broadcast, and the possession of an appropriate prime allows recovery of the key. However, that scheme revealed too much information about the primes of users in P . In this scheme, possession of the appropriate prime and the appropriate deciphering exponent allow recovery of K .

4.5.3 Eliminating the Trusted Center

The implementations given in Section 4.5.1 and Section 4.5.2 satisfy the parallel broadcasting model presented in Section 4.2. The trusted center T computes an encryption key K_P for the privileged set P , and creates a secure broadcast message using this key. Using only the secret that each user shares with T , allows recovery of the session key K , from the broadcast message.

Throughout this chapter, we have discussed the broadcasting of information from a trusted center T , to a privileged set P of users. We require trust in the center because the center shares a secret key with each user, i.e., just as in a symmetric cryptosystem. In other words, T can “impersonate” each user, since he too possesses the only secret information that the user has, i.e., the prime p_i . The center must be trusted not to divulge this secret prime to any unauthorized individual.

For applications where a trusted center is not desired, we note that the schemes of Section 4.5.1 and Section 4.5.2 can each be altered to remove the need for a trusted center T . In other words, in place of a prime p_i , each u_i can share a composite $n_i = p_{i1}p_{i2}$, the product of two primes, with T .

For the scheme of Section 4.5.2, we can use [RSA78] (see Section 1.1) rather than the encryption scheme of Pohlig and Hellman [Pohl78]. In other words, each user would share (possibly over an insecure channel) an RSA composite n_i with T , rather than simply a prime p_i . User u_i would then maintain the secrecy of the decryption exponent d_i such that $ed_i \equiv 1 \pmod{\phi(n)}$. The protocol would occur

as in Section 4.5.2, with n used in place of p . To ensure sufficient “wrap-around” and avoid the first attack mentioned in Section 1.2, c should be chosen to be $> t \lg(n_i)$ for the largest anticipated values of t and n_i . Even for a large t , this is not an unreasonable constraint.

The same trick can be used for the scheme of Section 4.5.1. Each user u_i would share a pair (y_i, n_i) with T . Notice that T must choose an appropriate y_i since T is unable to compute a quadratic non-residue, given a composite n_i [Gold84]. The protocol would proceed as in Section 4.5.1 except that n_i would be used in place of p_i .

4.6 Analysis of Parallel Broadcast Schemes

In this section, we emphasize the relevant properties of two methods of parallel broadcasting. We examine the model for broadcasting with information theoretic security (BITS) from Section 4.3.1, and more specifically, the scheme of Section 4.4.1 that satisfies this model. An analysis of the more practical of the two number theoretic broadcasting schemes from Section 4.5 is also presented.

The scheme from Section 4.4.1 that allowed us to meet the lower bound on the size of the broadcast message provides a nice link between the properties of secret sharing and broadcasting. Where secret sharing requires the interaction of users over a secure channel, broadcasting via secret sharing avoids interaction and allows distribution of the broadcast message over an insecure channel.

Since the session key K is being broadcast from one center T to a number of individual users, it is no surprise that the amount of computation is one-sided, i.e. T performs more work than an individual user. Recall that in the scheme of Section 4.4.1 (SS), the center performs $O(t^2)$ multiplications in Z_p^* to produce the broadcast message for t users. As a result of broadcasting the coefficients of the polynomial, each user need only perform $O(t)$ multiplications in Z_p^* . Using simple multiplication algorithms, these result in $O(t^2(\lg p)^2)$ and $O(t(\lg p)^2)$ respective bit operations.

For the Extended RSA scheme of Section 4.5.2 (ERSA), at most $2\lceil \lg N \rceil$ multiplications in Z_N^* are required (or $O(t^2(\lg p)^3)$ bit operations where we let $N \simeq p^t$ for some prime p , without loss of generality). This exceeds the amount of computation performed by T for the SS scheme. Recovery of the key by each user requires at most $2\lceil \lg p \rceil$ multiplications in Z_p^* or $O((\lg p)^2 \lg p)$ bit operations. This exceeds the computation performed by T in cases where $t < \lg p$.

At first glance, it may seem odd that the number theoretic scheme is comparable in the amount of key computation to the information theoretic scheme. However, recall that SS can only be used once unless the improvements from Section 4.4.3 are implemented. In Section 4.4.3, maintaining information theoretic security required that we increase the size of each user's share. This increase in share size would be proportional to the number of anticipated key distributions. On the other hand, ERSA can be used more than once, while maintaining the size of a user's share.

The size of the broadcast message is the same for both schemes. Notice that the share of each user (in both schemes) is proportional to a suitably chosen prime p . The broadcast message for establishing a key with t users is proportional to t times the size of a single user's share.

Therefore, ERSA would appear to be a more practical solution to parallel broadcasting. However, a search for other schemes satisfying the model of Section 4.2 is warranted.

Chapter 5

Conclusion

This thesis has examined several different methods for groups of more than two users to establish a common cryptographic key. Key agreement allows users to each contribute a single piece of information to the construction of a final key via exchanges over insecure channels. Several multi-party extensions of the Diffie-Hellman protocol were examined. Their relative performance and relation to an optimal multi-party Diffie-Hellman scheme were discussed. A framework for achieving an optimal scheme was presented. We described how to use this framework to construct a multi-party key agreement scheme.

Secret sharing requires users to physically “pool” together their shares to establish a key. We examined the two original secret sharing schemes and discussed methods of introducing reusability into secret sharing schemes.

We considered methods of broadcasting a message to users such that only a privileged subset of the users can establish a key K from this message. We began by examining schemes that use a serial construction of the broadcast message. A new framework that allows for the parallel construction of the broadcast message was introduced. A lower bound on the size of the broadcast message was achieved. Two schemes that use secret sharing to achieve a parallel construction of the broadcast message were presented. We proceeded to alter one of the schemes to allow it to reach the aforementioned lower bound. New schemes that use number theoretic problems to achieve a parallel broadcast construction were also introduced.

Future Work

Discovery of a multi-party key agreement protocol that extends the Diffie-Hellman two-party protocol would have made a nice addition to this thesis. Studying other functions besides exponentiation modulo a prime is one avenue of research.

To aid in this search, we note that there are several different types of Public Key Cryptosystems (cf. [Chor88])

1. PKCS's based on number-theoretic problems (e.g. [RSA78], [Elga85]) and
2. PKCS's based on NP-complete problems
 - the subset-sum (knapsack) problem (e.g. [Chor88]),
 - error correcting codes ([McEl78]),

while Diffie-Hellman is the most prominent key agreement protocol. Recall that the ElGamal cryptosystem [Elga85] is based on Diffie-Hellman. It would be interesting to attempt to use NP-complete problems (whose worst-case running times are exponential) to achieve multi-party key agreement.

Another area of potential research stems from different uses of the framework presented in Section 4.2. In Section 4.2, T creates a master key K_P associated with the privileged set P . While the message that is “locked” by this master key can be opened by that same key, it can also be opened using a single key from any one of the members of P . This differs from secret sharing where a set of user's keys are jointly required to open the lock. This arrangement whereby a single user (key) suffices to recover a message has many other potential applications, including access control.

Appendix A

Number Theory

The following sections introduce number theoretic concepts that are used in some of the previous chapters. See [Corm93, Chapter 33] for further background.

A.1 Computing Inverses

In this section, we present an algorithm for computing inverses modulo an integer m . Euclid's algorithm for computing the greatest common divisor (gcd) of two integers can be extended to what is known as the Extended Euclidian Algorithm (EEA) for computing the gcd of two integers. The EEA is extended because it also allows one to solve for x given equations of the form

$$ax \equiv b \pmod{m}. \quad (\text{A.1})$$

More specifically for our case is when $b = 1$, i.e., x will be the multiplicative inverse of a modulo m .

Euclid's algorithm for finding $\text{gcd}(a, m)$ for integers a and m uses the fact that $\text{gcd}(a, m) = \text{gcd}(m, a \bmod m)$.

```
gcd(a,m)
  if m=0 then
    return a
  else
    return gcd(m, a mod m)
```

The EEA finds the gcd of a and m , and integers x and y such that $ax + my = \gcd(a, m)$. When a and m are coprime, we are solving for x and y where $ax + my = 1$. This allows the determination of inverses, since $ax + my = 1$ can be rewritten as $ax \equiv 1 \pmod{m}$ (see (A.1)). Therefore, once we have x , we have found the inverse of a modulo m .

```

EEA(a,m):(x,y,d)
{On input a,m return x,y,d where ax + my = d = gcd(a,m).}
  if m=0 then
    return (1,0,a)
  (x',y',d') = EEA(a,m)
  x = y'
  y = x' - ⌊ $\frac{a}{m}$ ⌋y'
  d = d'
  return(x,y,d)

```

The EEA makes the same number of recursive calls as the gcd algorithm. At the last recursive call, $m = 0$ and EEA returns $(1, 0, a)$, i.e., $d = a = a(1) + m(0)$. When $m \neq 0$, EEA computes (x', y', d') such that $d' = \gcd(m, a \bmod m)$, or in other words $d' = mx' + (a \bmod m)y'$. Letting $d = d'$, as in the second last step, we obtain

$$\begin{aligned}
 d &= mx' + (a \bmod m)y' \\
 &= mx' + (a - \left\lfloor \frac{a}{m} \right\rfloor m)y' \\
 &= ay' + m(x' - \left\lfloor \frac{a}{m} \right\rfloor y')
 \end{aligned}$$

giving us the assignments $x = y'$ and $y = x' - \left\lfloor \frac{a}{m} \right\rfloor y'$ from EEA.

A.2 Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) provides a method for finding a solution to a series of equations, each having a distinct modulus. The proof of the theorem is constructive, i.e., it provides a method of constructing such a solution.

Theorem A.1 *Let $n_1, n_2, \dots, n_t$ be pairwise relatively prime integers and let $N = n_1 n_2 \cdots n_t$. For any integers $r_1, r_2, \dots, r_t$, the series of equations*

$$\begin{aligned} r_1 &\equiv C \pmod{n_1} \\ r_2 &\equiv C \pmod{n_2} \\ &\vdots \\ r_t &\equiv C \pmod{n_t} \end{aligned}$$

has a unique (in this range) solution $0 \leq C \leq N - 1$.

Proof:

For $1 \leq i \leq t$, $\gcd(n_i, N/n_i) = 1$, as a result of the fact that n_i does not divide (N/n_i) . Therefore, there are unique y_i such that

$$\left(\frac{N}{n_i}\right) y_i \equiv 1 \pmod{n_i}.$$

From Appendix A.1, we know that the y_i exist, and we know how to find them. It is also true that $\gcd(n_j, N/n_i) = n_j \neq 1$, $i \neq j$. Therefore

$$\left(\frac{N}{n_i}\right) y_i \equiv 0 \pmod{n_j}.$$

This is a result of the fact that $n_j | (N/n_i)$, $i \neq j$. We can construct C as follows

$$C = \sum_{i=1}^t \left(\frac{N}{n_i}\right) y_i r_i \pmod{N}. \quad (\text{A.2})$$

Notice that for each n_i , we have

$$\begin{aligned} C &\equiv \left(\frac{N}{n_i}\right) y_i r_i \pmod{n_i} \\ &\equiv r_i \pmod{n_i}. \end{aligned}$$

□

References

- [Berk92] Berkovits, S., "How to Broadcast a Secret", *Advances in Cryptology: Proceedings of EUROCRYPT '91*, Springer-Verlag, 1992, pp.536-541.
- [Blak79] Blakley, G., "Safeguarding Cryptographic Keys", *Proceedings NCC*, AFIPS Press, Vol. 48, 1979, pp. 313-317.
- [Blak80] Blakley, G., "One-time pads are Key Safeguarding Schemes, not Cryptosystems: Fast Key Safeguarding Schemes (Threshold Schemes) Exist", *Proceedings of IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1980, pp.108-113.
- [Blak92] Blakley, B., Blakley, G., Chan, A., Massey, J., "Threshold Schemes With Disenrollment", *Advances in Cryptology: Proceedings of CRYPTO '92*, Springer-Verlag.
- [Blun93] Blundo, C., Cresti, A., De Santis, A., Vaccaro, U., "Fully Dynamic Secret Sharing Schemes", *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag.
- [Burm94] Burmester, M., Desmedt, Y., "A Secure and Efficient Conference Key Distribution System", to appear in *Advances in Cryptology: Proceedings of Eurocrypt '94*, Springer-Verlag.
- [Bras88] Brassard, G., "Modern Cryptology-A Tutorial", *Lecture Notes in Computer Science*. Springer-Verlag, 1988.

- [Bric89] Brickell, E., "Some Ideal Secret Sharing Schemes", *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol.6, 1989, pp. 105-113.
- [Capo92] Capocelli, R., De Santis, A., Gargano, L., Vaccaro, U., "On the Size of Shares for Secret Sharing Schemes", *Advances in Cryptology: Proceedings of CRYPTO '91*, Springer-Verlag, 1992, pp.101-113.
- [Chan92] Chang, C.C., Hwang, S.J., "A Secure Broadcasting Scheme Based on Discrete Logarithms", *Control and Computers*, Vol.20, No.2, 1992, pp.49-53.
- [Chio89] Chiou, G.H., Chen, W.T., "Secure Broadcasting Using the Secure Lock", *IEEE Transactions on Software Engineering*, Vol.15, No.8, August 1989, pp.929-934.
- [Chor88] Chor, B., Rivest, R., "A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields", *IEEE Transactions on Information Theory*, Vol.IT-34, No.5, Sept.1988, pp.901-909.
- [Corm93] Cormen, T., Leiserson, C., Rivest, R., *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 1993.
- [DeLa84] DeLaurentis, J., "A Further Weakness in the Common Modulus Protocol for the RSA Cryptalgorithm", *Cryptologia*, Vol.8, No. 3, July 1984, pp. 253-259.
- [Denn83] Denning, D.E., *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1983.
- [Diff76] Diffie, W., Hellman, M., "New Directions in Cryptography", *IEEE Transactions on Information Theory*, IT-22(6), November 1976, pp.644-654.
- [Diff88] Diffie, W., "The First Ten Years of Public-Key Cryptography", *Proceedings of the IEEE*, Vol.76, No.5, 1988, pp.560-577.

- [Elga85] ElGamal, T., "A public key cryptosystem and signature scheme based on discrete logarithm", *IEEE Transactions on Information Theory*, Vol.IT-31, 1985, pp.469-472.
- [Fiat93] Fiat, A., Naor, M., "Broadcast Encryption", *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag.
- [Gold84] Goldwasser, S., Micali, S., "Probabilistic Encryption", *Journal of Computer and System Sciences*, Vol.28, 1984, pp.270-299.
- [Gong94] Gong, L., "Authentication, Key Distribution, and Secure Broadcast in Computer Networks Using No Encryption or Decryption", to appear in *2nd Annual ACM Conference on Computer Security*, Fairfax, VA. 1994, (also appears as *Technical Report SRI-CSL-94-08*, Computer Science Laboratory, SRI International, Menlo Park, CA, May, 1994.)
- [Gopa82] Gopal, J., Jaffe, M., "Point-to-multipoint Communication over Broadcast Links", *IEEE Transactions on Communications*, COM-32(9), 1982, pp.1034-1044.
- [Hast85] Hastad, J., "On Using RSA with Low Exponent in a Public Key Network", *Advances in Cryptology: Proceedings of CRYPTO '85*, Springer-Verlag, 1986, pp.403-408.
- [Inge82] Ingemarsson, I., Tang, D., Wong, C., "A Conference Key Distribution System", *IEEE Transactions on Information Theory*, Vol. IT-28, No.5, Sept. 1982, pp.714-720.
- [Juma90] Jumaie, G., *Relative Information: Theories and Applications*, Springer-Verlag, Berlin, 1990.
- [Just94] Just, M., Kranakis, E., Krizanc, D., van Oorschot, P., "On Key Distribution via True Broadcasting", to appear in *2nd Annual ACM Conference on*

- Computer Security*, Fairfax, VA. 1994, (earlier version appears as "Broadcasting Session Keys", *Technical Report TR-293*, Carleton University, School of Computer Science, Feb. 1994.)
- [Kraw93] Krawczyk, H., "Secret Sharing Made Short", *Advances in Cryptology: Proceedings of CRYPTO '93*, Springer-Verlag.
- [Laih89] Lai, C., Lee, J., Harn, L., "A New Threshold Scheme and its Applications in Designing the Conference Key Distribution Cryptosystem", *Information Processing Letters*, Vol.32, 1989, pp.95-99.
- [Lin92] Lin, C.H., Chang, C.C., Lee, R.C., "A Conference Key Broadcasting System Using Sealed Locks", *Information Systems*, Vol.17, No.4, 1992, pp.323-328.
- [McEl78] McEliece, R., "A Public Key Cryptosystem Based on Algebraic Coding Theory", *DSN Progress Report 42-44*, Jet Propulsion Laboratory, 1978, pp.114-116.
- [DES77] National Bureau of Standards, "Data Encryption Standard", *Federal Information Processing Standard*, U.S. Department of Commerce, FIPS PUB 46, Washington, DC, 1977.
- [Pohl78] Pohlig, S., Hellman, M., "An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance", *IEEE Transactions on Information Theory*, Vol.IT-24, 1978, pp.106-110.
- [Rabi93] Rabin, M., Sherman, A., "Associative One-Way Functions: A New Paradigm for Secret-Key Agreement and Digital Signatures", *Technical Report CS-TR-3183/UMIACS-TR-93-124*, University of Maryland College Park, July 1993, (also *Technical Report TR-CS-93-18*, Computer Science Department, University of Maryland, Baltimore County, November, 1993).
- [Rabi78] Rabin, M.O., "Digital Signatures and Public-Key Functions as Intractable as Factorization", *Technical Report MIT/LCS/TR-212*, M.I.T., 1978.

- [Rab80] Rabin, M., "Probabilistic Algorithms in Finite Fields", *SIAM Journal on Computing*, Vol.9, No.2, 1980, pp.273-280.
- [Ries85] Riesel, H., *Prime Numbers and Computer Methods for Factorization*, Birkhäuser Inc., Boston, MA, 1985.
- [RSA78] Rivest, R., Shamir, A., Adleman, L. "A Method for Obtaining Digital Signatures and Public-Key *Communications of the ACM*, Vol.21, 1978, pp.120-126.
- [Ruep94] Rueppel, R., van Oorschot, P., "Modern Key Agreement Techniques", *Computer Communications Journal*, July 1994.
- [Sham79] Shamir, A., "How to Share a Secret", *Communications of the ACM*, Vol.22, No.11, November 1979, pp.612-613.
- [Shan49] Shannon, C., "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, Vol.28, 1949, pp.656-715.
- [Sher94] Sherman, A., *personal communication*, July, 1994.
- [Simm83] Simmons, G., "A Weak Privacy Protocol Using the RSA Crypto Algorithm", *Cryptologia*, Vol. 7, No.2, April 1983, pp. 180-182.
- [Simm91] Simmons, G., "An Introduction to Shared Secret and/or Shared Control Schemes and their Application", *Contemporary Cryptology*, IEEE Press, 1991, pp.441-497.
- [Stee88] Steer, D., Strawczynski, L., Diffie, W., Wiener, M., "A Secure Audio Teleconference System", *Advances in Cryptology: Proceedings of CRYPTO '88*, Springer-Verlag, 1988, pp.520-528.
- [Wegm81] Wegman, M., Carter, J., "New Hash Functions and Their Use in Authentication and Set Equality", *Journal of Computer and System Sciences*, Vol.22, 1981, pp.265-279.

- [Zhen94] Zheng, Y., Hardjono, T., Seberry, J., "Reusing Shares in Secret Sharing Schemes", *The Computer Journal*, Vol.37, No.3, 1994, pp.199-205.

END

3 1-05-95

FIN