UNIVERSITY OF CALIFORNIA, SAN DIEGO

Design and Analysis of Secure Encryption Schemes

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Michel Ferreira Abdalla

Committee in charge:

Professor Mihir Bellare, Chairperson
Professor Yeshaiahu Fainman
Professor Adriano Garsia
Professor Mohan Paturi
Professor Bennet Yee

2001

The dissertation of Michel Ferreira Abdalla is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2001

DEDICATION

To my father (*in memorian*)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and guidance of my advisor, Professor Mihir Bellare, to whom I am deeply grateful. I would like to thank him for introducing me to the area of cryptography and for the countless discussions on the topic. It has been a privilege working with him and having him as a friend.

I would also like to thank Professor Yeshaiahu Fainman, Professor Adriano Garsia, Professor Mohan Paturi, and Professor Bennet Yee, for agreeing to be part of my thesis committee.

Most of my research has been done in collaboration with many people. I have been fortunate to have worked with Jee Hea An, Mihir Bellare, Sasha Boldyreva, Walfredo Cirne, Anand Desai, Leslie Franklin, Matt Franklin, Keith Marzullo, Sara Miner, Meaw Namprempre, David Pointcheval, Leonid Reyzin, Phil Rogaway, Mike Semanko, Yuval Shavitt, Abdallah Tabbara, Bogdan Warinschi and Avishai Wool. A significant part of this thesis is based on joint work with Mihir Bellare, Phillip Rogaway, Yuval Shavitt, and Avishai Wool. I thank them for allowing me to include results based on these joint works.

I would also like to use this opportunity to thank my friend and previous advisor, Otto Duarte, and my friends, Celio Albuquerque, Marcio Faerman, Patricia Pinto, and Ana Velloso, for encouraging me to pursue my Ph.D. studies at UCSD. Celio and Marcio were particularly helpful in assisting me to settle down in my new place. I only had a weekend between my arrival in the United States and the beginning of classes to furnish my new apartment and I would not have made it without their help. I will always be grateful to all of them.

My special thanks to all my friends in the Cryptography and Security Lab and in the Computer Science department: Jee Hea An, Andre Barroso, Mihir Bellare, Sasha Boldyreva, Henri Casanova, Walfredo Cirne, Matthew Dailey, Holly Dail, Sashka Davis, Anand Desai, Marcio Faerman, Leslie Franklin, Alejandro Hevia, Matt Hohlfeld, Flavio Junqueira, Barbara Kreaseck, Daniele Mic-

ciancio, Sara Miner, Meaw Namprempre, Graziano Obertelli, Adriana Palacio, Mike Semanko, Alan Su, Renata Teixeira, Bogdan Warinschi, Bennet Yee, Bianca Zadrozny, and Dmitrii Zagorodnov.

I thank my mother Nadyr Abdalla, my sister Patricia Abdalla, and my girlfriend Stacie Ngo for their love and support.

I thank CAPES for providing the financial support for my Ph.D. studies.

VITA

June 27, 1970	Born, Rio de Janeiro, Brazil
1993	B.S. Universidade Federal do Rio de Janeiro, Brazil
1993–1994	Indice Desenvolvimento de Sistemas
1996	M.S., COPPE/UFRJ, Brazil
1996–2001	Research Assistant, University of California, San Diego
2001	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

M. Abdalla, M. Bellare and P. Rogaway. "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES." Topics in Cryptology – CT-RSA 2001, Lecture Notes in Computer Science 2020, pp. 143–158, Springer-Verlag, D. Naccache ed., 2001.

M. Abdalla, S. Miner and C. Namprempre. "Forward Security in Threshold Signature Schemes." Topics in Cryptology – CT-RSA 2001, Lecture Notes in Computer Science 2020, pp. 441–456, Springer-Verlag, D. Naccache ed., 2001.

M. Abdalla and M. Bellare. "Increasing the Lifetime of a Key: a Comparative Analysis of the Security of Re-keying Techniques." Advances in Cryptology – ASIACRYPT 2000, Lecture Notes in Computer Science 1976, pp. 546–559, T. Okamoto ed., Springer-Verlag, 2000.

M. Abdalla and L. Reyzin. "A New Forward-secure Digital Signature Scheme." Advances in Cryptology – ASIACRYPT 2000, Lecture Notes in Computer Science 1976, pp. 116–129, T. Okamoto ed., Springer-Verlag, 2000.

M. Abdalla, Y. Shavitt and A. Wool. "Key Management for Restricted Multicast Using Broadcast Encryption." IEEE/ACM Transactions on Networking, Vol. 8, Number 4, pp. 443–454, August, 2000.

M. Abdalla, Y. Shavitt and A. Wool. "Towards Making Broadcast Encryption Practical." Financial Cryptography'99, Lecture Notes in Computer Science 1648, pp. 140-157, M. Franklin ed., Springer-Verlag, 1999.

M. Abdalla, W. Cirne, L. Franklin and A. Tabbara. "Security Issues in Agent Based Computing." in Proc. of the 15th Brazilian Symposium on Computer Networks, Campinas, Brazil, May 1997.

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Cryptography and Network Security.
Professor Mihir Bellare and Bennet Yee

Studies in Electrical and Computer Engineering.
Professor Yeshaiahu Fainman

Studies in Mathematics.
Professor Adriano Garsia

Studies in Complexity and Theory.
Professors Mohan Paturi

ABSTRACT OF THE DISSERTATION

Design and Analysis of
Secure Encryption Schemes

by

Michel Ferreira Abdalla
Doctor of Philosophy in Computer Science
University of California, San Diego, 2001
Professor Mihir Bellare, Chair

In this thesis, we design, and analyze the security of, several encryption schemes both in the private-key and public-key setting. Our main goal is to provide schemes for which we can provide theoretical proofs of security, but which are also efficient and practical.

We begin by describing a new public-key encryption scheme based on the Diffie-Hellman problem, called DHIES. The main goal of DHIES is to not only be as efficient as the ElGamal encryption scheme, but to also provide security against chosen-ciphertext attacks. DHIES is a Diffie-Hellman based scheme that combines a private-key encryption method, a message authentication code, and a hash function, in addition to number-theoretic operations. The proofs of security are based on the assumption that the underlying symmetric primitives are secure and on appropriate assumptions about the Diffie-Hellman problem. Our proofs are in the standard model; no random-oracle assumption is required. DHIES is now in the draft standards of ANSI X9.63 [6] and IEEE P1363a [50] and in the corporate standard SECG [71].

Next, we study re-keyed encryption schemes. These are schemes in which shared keys are not used directly to encrypt messages, but rather used as a master key to derive sub-keys, which are then used to encrypt messages. This is a

commonly employed paradigm in computer security systems, about whose security benefits users appear to have various expectations. In this thesis, we provide concrete security analyses of various re-keying mechanisms and their usage. We show that re-keying does indeed "increase" security, effectively extending the lifetime of the master key and bringing significant, provable security gains in practical situations.

We also study the problem of secure encryption in the broadcast model, where a broadcast center wants to communicate securely with a set of users (the target set) over an insecure broadcast channel.

Chapter I

Introduction

A fundamental problem in cryptography is how to communicate securely over an insecure channel, which might be controlled by an adversary. This problem has for a long time fascinated people and has become even more important with the proliferation of computers and communication networks, such as the Internet. From simple message exchanges among friends, to purchases made with a credit card, to the transmission of sensitive documents, these communication systems are now being used for many different purposes.

The problem of secure communication has many different flavors depending on the security aspect in which we are interested and on how powerful adversaries can be. In this thesis, we are mainly concerned with the privacy aspect of the communication.

I.A Encryption

Encryption is certainly one of the most fundamental problems in cryptography and it is usually what it comes to mind when one talks about cryptography. The latter, however, encompasses many other areas, including, but not limited to, message authentication, digital signatures, identification protocols, and zero-knowledge protocols.

I.A.1 Background

The two main parties involved in a encryption scheme are the sender and the receiver. Whenever a sender wants to transmit a message, called the *plaintext*, to the receiver, it first converts it to an encrypted form, called the *ciphertext*, and then sends it to the receiver. The method used to convert the plaintext into a ciphertext is called the *encryption* algorithm. Upon receiving the ciphertext, the receiver uses a *decryption* algorithm to recover the original plaintext from the ciphertext it received.

The goal of an encryption algorithm is simple —to allow the sender and receiver to communicate *privately* over an insecure communication channel, possibly under the control of an adversary. Such an adversary is usually able to eavesdrop, and even alter in some cases, the communication that is taking place between the sender and receiver. Hence, it becomes clear that in order to achieve privacy in communication, the receiver needs to have some sort of information advantage with respect to the adversary or else the adversary could recover the original plaintext from ciphertext in the same way the receiver does, since it has access to all information being exchanged over the communication channel. The information advantage of the receiver could take several forms, but it usually involves the knowledge of a secret not known by the adversary. This secret could be, for example, the knowledge of the encryption and decryption algorithms themselves or part of their inputs. The case in which the encryption and decryption algorithms themselves are kept hidden from the adversary is known as obscurity and is particularly interesting in cases where message lengths are small and the amount of resources available to sender and receiver is limited. For instance, two people willing to privately exchange fixed-length messages with one another could agree in advance on a fixed but random way of permuting the letters in each message. The receiver, knowing the way letters were permuted in a message, can recover the original plaintext from the ciphertext by simply computing the inverse permutation on the ciphertext. An adversary, on the other hand, would not be

able to recover the original plaintext since it is not aware of the way in which the letters were permuted. This problem with this approach is that is not efficient when messages are long as the amount of secret information that each party needs to store can increase significantly.

A method more commonly used nowadays is to hide part of the input of the encryption and decryption algorithms and not the algorithms themselves. This input is typically called the *key* and will be fixed within each instantiation of the encryption scheme. The keys used by the encryption and decryption algorithms are called, respectively, the *encryption key* and the *decryption key*. The decryption key should always be kept secret and, for this reason, is also called the *secret key*. The encryption key, on the other hand, might be either secret, as in the case of private-key encryption, or mathematically related to the decryption key, as in the case of public-key encryption. Because the encryption and decryption keys are usually equal in private-key encryption and different in public-key encryption, these two are also referred to as symmetric and asymmetric encryption, respectively. The algorithm used to generate the keys in either the symmetric and asymmetric encryption schemes is called the *key generation* algorithm.

I.A.2 Perfect Privacy

Up to now, we have mentioned that some information advantage is necessary in order to achieve private communication between two parties, but we have not defined what privacy means. It turns out that providing such a definition is not as straight-forward as one may think and can be very tricky sometimes. The first formal definition for what it is now known as *perfect privacy* was given by Shannon [73]. It describes privacy in terms of information entropy and it says that the amount of entropy in a plaintext given the ciphertext should be the same as amount of entropy in the plaintext itself. In other words, no information about the plaintext should be leaked through the ciphertext. Shannon was also the first to provide a symmetric encryption scheme that *provably* meets this formal notion

of privacy. In this scheme, called the *one-time pad*, both the sender and receiver share a secret key whose length is at least that of the message being exchanged. To encrypt a message, the sender simply computes the bit-wise x-or[1] of the secret key with the message to send it to receiver. The original plaintext can then be recovered from the ciphertext by simply computing the bit-wise x-or of the latter with the secret key. The main advantage of this scheme is that it is *impossible* for an adversary to extract any information about the plaintext from the ciphertext. Its main disadvantage is the fact that it requires a secret key at least as long the message being encrypted. In fact, as it was also shown by Shannon, this condition is not only sufficient but also necessary to achieve perfect privacy.

I.A.3 Modern cryptography

A scheme meeting the notion of perfect privacy is secure in an *information-theoretic* sense since it is *impossible* for an adversary to extract any information about the plaintext from the ciphertext. A different approach adopted in modern cryptography is to define privacy in terms of *infeasibility* rather than *impossibility* as adversaries are computationally bounded in practice. In this *computational-complexity* approach, it is permitted for a ciphertext to leak information about the plaintext as long as this information cannot be *efficiently* computed. Here, efficiently means computable in polynomial time. An example of a definition for privacy in the *computational-complexity* sense is that of indistinguishability of encryptions, given by Goldwasser and Micali [44], which says that no adversary should be to efficiently distinguish between the encryption of two plaintexts with non-negligible probability of success. In that work, they show that, if a scheme meets the above notion, then no adversary can efficiently compute any information about the plaintext from the ciphertext.

There are many advantages in adopting the *computational-complexity* approach. Here we list two of them. First, the key length can be much shorter

[1]The x-or of two bits b_0 and b_1 is simply $b_0 + b_1$ modulo 2.

than those in *information-theoretic* approach. In particular, the key length can be smaller than the message length. For example, by using a pseudorandom generator on a random seed, one could generate arbitrarily long (but polynomial on the length of the seed) pseudorandom strings to be used as the secret key in the Shannon's one-time pad scheme presented above. Second, it allows for the existence of public-key encryption. The latter is not possible in the information-theoretic framework since it is possible, even if not efficiently, for an adversary to compute the secret (decryption) key from the public (encryption) key.

I.A.4 Public-key Encryption

The notion of public-key cryptography, introduced by Diffie and Hellman [32], is perhaps one of main innovations in modern cryptography. In contrast to private-key cryptography, in which two parties need to share a secret in advance, public-key cryptography enables secure communication between two parties without requiring previously established secrets.

In public-key cryptography, each user has a pair of related keys associated to it, a *public key* and a *secret key*. While the public key is made public to all parties, including the adversary, and secret key should only be known by the user to which it is associated. In the particular case of public-key encryption, the public and secret key become, respectively, the encryption and decryption keys. In order to send a message to a user, one can simply encrypt the message using the public key associated to that user. Whenever a user receives a ciphertext, supposedly encrypted with its public key, it decrypts it using its secret key.

The first public-key encryption scheme was proposed by Rivest, Shamir, and Adleman [69] and it is now known as the RSA cryptosystem.

I.A.5 Broadcast Encryption

Even though the primary use of encryption is for secure point-to-point communication, this is certainly not its only use. With the increasing number of

users connected to large communication networks, such as the Internet, the number of applications in which encryption is being used is growing significantly. One such example is broadcast encryption.

The main parties in a broadcast encryption scheme are the broadcast center and a set of users, all connected to an insecure broadcast channel. The goal of a broadcast encryption scheme is to allow the broadcast center to securely communicate with a subset of all users. The subset of users will vary according to the particular application. For example, in satellite/cable pay TV pplications, the subset of users can be those users who subscribed and paid for a particular service or program. Because the number of users can be incredibly large, the scalability of solution is crucial in the design of these schemes.

Other examples of applications in which encryption is often used are threshold encryption and key distribution.

I.A.6 Provable Security

A very common paradigm used in modern cryptography for designing protocols is to build on top of existing cryptographic primitives, whose security properties are well-defined and well-understood. Such cryptographic primitives usually solve very specific problems and are hard to find in practice. The security of these primitives, however, is not usually the main part of the security analysis of a protocol. Most often, cryptographic protocols are broken not because the security of the basic primitives got broken but because of the way in which these primitives are used. That is exactly where provable security comes into play. It enables us to provide a greater assurance of security, by proving that the scheme meets formally defined objectives under a given model and the assumption that the cryptographic primitives are secure.

A cryptographic scheme S based on a primitive P is said to be *provably secure* if the security of P has been demonstrated to imply the security of S. More precisely, we use this phrase when someone has formally defined the goals G_P

and G_S for some primitive P and scheme S, respectively; and then has proven that the existence of an adversary A_S who breaks scheme S, in the sense of violating G_S, implies the existence of an adversary A_P who breaks primitive P, in the sense of violating G_P.

What provable security means is that as long as we are ready to believe that P is secure, then there are no attacks on S. This obviates the need to consider any specific cryptanalytic attacks on S.

Examples of candidates that can be used as basic cryptographic primitives include pseudorandom generators and block ciphers as well as number-theoretic functions such as the RSA one-way function.

I.A.7 Concrete Security

Although provable security can tell us when a scheme is secure, it does not necessarily say how secure a scheme really is. For instance, when comparing the security of different schemes, it might be helpful to know how tightly or loosely related the security of these schemes is with respect to the security of their underlying primitives. A more quantitative approach to provable security may be more desirable in these situations.

This concrete and quantitative approach to provable security is usually called *practice-oriented provable security* and was developed by Bellare and Rogaway [15, 16]. The general idea is as follows. Let S be a cryptographic scheme which makes use of a primitive P, and let A_S be an adversary which attacks S. When proving the security of S, one should convert the adversary A_S attacking S into an adversary A_P which attacks the primitive P. Ideally, A_P should use the same computational resources as A_S and, with this investment in resources, A_P should be just as successful in attacking the primitive P as A_S was successful in attacking the cryptographic scheme S. This way, "practical" attacks on P imply practical attacks on S, and so the assumed *absence* of practical attacks on P implies the absence of practical attacks on S.

To quantify how close we come to the ideal, we define the success probability of A_P attacking P and the success probability of A_S attacking S, and then we give concrete formulas that show how A_P's computational resources and success probability depend on A_S's computational resources and success probability. These formulas measure the demonstrated security. By giving explicit formulas, we make statements which are more precise than those that are given in doing asymptotic analyses of reductions.

I.B Contributions

In this thesis, we design and analyze the security of several encryption schemes both in the symmetric and asymmetric setting. Our main goal is to provide schemes that are theoretically interesting, but are also efficient and practical.

We start by describing a new public-key encryption scheme based on the Diffie-Hellman problem, which we call DHIES. It is a simple extension of the ElGamal encryption scheme and is now in the draft standards of ANSI X9.63 [6] and IEEE P1363a [50] and in the corporate standard SECG [71]. The scheme is as efficient as ElGamal encryption, but has stronger security properties. Furthermore, these security properties are proven to hold under appropriate assumptions on the underlying primitive. DHIES is a Diffie-Hellman based scheme that combines a symmetric encryption method, a message authentication code, and a hash function, in addition to number-theoretic operations, in a way which is intended to provide security against chosen-ciphertext attacks. The proofs of security are based on the assumption that the underlying symmetric primitives are secure and on appropriate assumptions about the Diffie-Hellman problem. The latter are interesting variants of the customary assumptions on the Diffie-Hellman problem, and we investigate relationships among them, and provide security lower bounds. Our proofs are in the standard model; no random-oracle assumption is required. These results are based on previous joint work with Mihir Bellare and Phillip Rogaway [2, 3].

Next, we study re-keyed encryption schemes. These are schemes in which shared keys are not used directly to encrypt messages, but rather used as a master key to derive sub-keys, which are then used to encrypt messages. This is a commonly employed paradigm in computer security systems, about whose security benefits users appear to have various expectations. Yet the security of these methods has not been systematically investigated. In this thesis, we provide concrete security analyses of various re-keying mechanisms and their usage. We show that re-keying does indeed "increase" security, effectively extending the lifetime of the master key and bringing significant, provable security gains in practical situations. We quantify the security provided by different re-keying processes as a function of the security of the primitives they use, thereby enabling a user to choose between different re-keying processes given the constraints of some application. These results are based on a previous joint work with Mihir Bellare [1].

Finally, we examine the problem of secure communication in the broadcast model. In this model, a user, called the broadcast center, wants to communicate securely with a set of users (the target set) over an insecure broadcast channel. This problem occurs in two application domains: satellite/cable pay TV, and the Internet MBone. In these systems, the parameters of major concern are the number of key transmissions, and the number of keys held by each receiver. In the Internet domain, previous schemes suggest building a separate key tree for each multicast program, thus incurring a setup cost of at least $k \log k$ per program for target sets of size k. In the pay TV domain, a single key structure is used for all programs, but known theoretical bounds show that either very long transmissions are required, or that each receiver needs to keep prohibitively many keys.

We propose broadcast encryption schemes that are targeted at both domains. Our schemes maintain a single key structure that requires each receiver to keep only a logarithmic number of establishment keys for its entire lifetime, while admitting low numbers of transmissions. In order to achieve these goals, we allow a controlled number of users outside the target set to occasionally receive the mul-

ticast. This relaxation is appropriate for many scenarios in which the encryption is used to force consumers to pay for a service, rather than to withhold sensitive information. For this purpose, we introduce *f-redundant* establishment key allocations, which guarantee that the total number of recipients is no more than f times the number of intended recipients. We measure the performance of such schemes by the number of key transmissions they require, by their redundancy f, and by the probability that a user outside the target set will be able to decrypt the multicast. We prove a new lower bound, present several new establishment key allocations, and evaluate our schemes' performance by extensive simulation. These results are based on previous joint work with Yuval Shavitt and Avishai Wool [4, 5].

Chapter II

Efficient public-key encryption schemes

Over the last decade, several schemes for public-key encryption have been proposed in the literature. Most of them, however, either lack a proof of security or are very inneficient. To overcome this problem and be able to create efficient, provably secure, encryption schemes, some researchers make use of a idealized model, the Random Oracle model, in which hash functions are modeled as public random oracles. A proof of security in this enriched model is then seen as an (heuristic) security assurance in the standard model, based on the thesis that the instantiation of these oracles by cryptographic hash functions would not compromise the overall security. More recently, however, some concerns about the use of the RO model have been raised and the construction of efficient schemes which can be proven secure in the standard model has become of major interest.

This chapter describes a Diffie-Hellman based encryption scheme, DHIES (formerly named DHES and DHAES), which is now in several (draft) standards. The scheme is as efficient as ElGamal encryption, but has stronger security properties. Furthermore, these security properties are proven to hold under appropriate assumptions on the underlying primitive. DHIES is a Diffie-Hellman based scheme that combines a symmetric encryption method, a message authentication

code, and a hash function, in addition to number-theoretic operations, in a way which is intended to provide security against chosen-ciphertext attacks. The proofs of security are based on the assumption that the underlying symmetric primitives are secure and on appropriate assumptions about the Diffie-Hellman problem. The latter are interesting variants of the customary assumptions on the Diffie-Hellman problem, and we investigate relationships among them, and provide security lower bounds. Our proofs are in the standard model; no random-oracle assumption is required. These results are based on previous joint work with Mihir Bellare and Phillip Rogaway [2, 3].

II.A Introduction

This chapter describes a method for encrypting strings using the Diffie-Hellman assumption. We are concerned with the "details" of Diffie-Hellman based encryption — how a message should be "packaged" in order to best exploit the group operations (e.g., modular exponentiation) which are at the core of a Diffie-Hellman based encryption.

The method we suggest is called DHIES, standing for "Diffie-Hellman Integrated Encryption Scheme". It is a simple extension of the ElGamal encryption scheme and is now in the draft standards of ANSI X9.63 and IEEE P1363a [6, 50] and in the corporate standard SECG [71]. The scheme was formerly known as DHES and as DHAES. It is all the same scheme.

DHIES uses symmetric encryption, message authentication, and hashing. This may seem like a lot of cryptography beyond the group operation, but it is exactly this additional cryptography which ensures, by and large, that we get our security guarantees.

The security analysis of DHIES requires some interesting new variants of the Diffie-Hellman assumption. We look at relationships among these notions, and we prove a complexity lower bound, in the generic model, about one of them.

BACKGROUND. DHIES is designed to be a natural extension of the ElGamal scheme, suitable in a variety of groups, and which enhanced ElGamal in a couple of ways important to cryptographic practice. First, the scheme needs to provide the capability of encrypting arbitrary bit strings (ElGamal requires that message be a group element). And second, the scheme should be secure against chosen-ciphertext attack (ElGamal is not). The above two goals have to be realized without increasing the number of group operations for encryption and decryption, and without increasing key sizes relative to ElGamal. Within these constraints, we want to provide the best possible provable-security analysis. But efficiency and practicality of the scheme should not be sacrificed in order to reduce assumptions.

The approach above is somewhat in contrast to related schemes in the literature. More typical is to fix an assumption and then strive to find the lowest cost scheme which can be proven secure under that assumption. Examples of work in this style are that of Cramer and Shoup [30] and that of Shoup [76], who start from the decisional Diffie-Hellman assumption, and then try to find the best scheme they can that will resist chosen-ciphertext attack under this assumption. In fact, the latter can also be proved secure in the random oracle model based on the weaker computational Diffie-Hellman assumption. These schemes are remarkable, but their costs are about double that of ElGamal, which is already enough to dampen some practical interest. A somewhat different approach was taken by Fujisaki and Okamoto [39], starting from weaker asymmetric and symmetric schemes to construct a stronger hybrid asymmetric scheme. Their scheme can be quite practical, but the proof of security relies heavily on the use of random oracles.

The DHIES scheme uses a hash function. In [17], a claim is made that DHIES should achieve plaintext awareness if this hash function is modeled as a public random oracle and one assumes the computational Diffie-Hellman assumption. In fact, technical problems would seem to thwart any possibility of pushing through such a result.

OUR APPROACH. DHIES is a very "natural" scheme. (See Section II.C

for its definition.) The method follows standard ideas and practice. Intuitively, it is secure. Yet it seems difficult to prove security under existing assumptions about the Diffie-Hellman problem.

This situation seems to arise frequently. It seems often to be the case that we think certain methods are good, but we don't know how to prove that they are good starting from "standard" assumptions. We suggest that what we are seeing with DHIES is a manifestation of hardness properties of Diffie-Hellman problems which just haven't been made explicit so far.

In this chapter, we capture some of these hardness properties as formal assumptions. We will then show how DHIES can then be proven secure under these assumptions. Then we further explore these assumptions by studying their complexity in the generic model [75], and by studying how the assumptions relate to one other.

Related work. As we have indicated, the DHIES scheme first appears in [17]. No proof appears in that work. It was suggested that a proof of plaintext awareness [15, 9] could be achieved under the random-oracle model. However, no such proof has appeared, and technical difficulties would seem to bar it.

DHIES is now embodied in three (draft) standards [6, 50, 71]. All of these assume an elliptic curve group of prime order. To harmonize our work with those standards, and to simplify complexity assumptions, we shall assume the the underlying group in which we work has prime order. When working with a group whose order is not prime a minor change can be made to the protocol so that it will still be correct. Namely, the value g^u should be fed into the hash function H.

Zheng and Seberry [83] have proposed an ElGamal-based scheme that uses universal one-way hash functions. Security of their scheme is not supported by proofs in the reductionist sense of modern cryptography. Lim and Lee [55] have pointed out that in some of the cryptosystems proposed in [83], the method of adding authentication capability may fail just under known plaintext attacks. A submission to IEEE P1363a based on [83] has been made by Zheng [82].

Another contemporaneous suggestion was proposed by Johnson, Matyas, and Peyravian [51]. Assume that the message M already contains some redundancy (e.g., some number of fixed bits) and unpredictability (e.g., random bits have been embedded in M). Then to asymmetrically encrypt M, [51] suggest to subject it to 4 rounds of a Feistel network based on a function H, thereby obtaining a new string M'. Encrypt, using an arbitrary encryption primitive, an arbitrary piece of M'. It is plausible that if H is modeled as a random function then the above approach can be proven sound.

Cramer and Shoup describe an encryption scheme based on the decisional Diffie-Hellman problem which achieves provable security against adaptive chosen-ciphertext attack [30]. They prove their scheme secure under the decisional Diffie-Hellman assumption (and a collision-intractable hash function), or, in the random-oracle model, under the ordinary Diffie-Hellman assumption [74]. Their scheme is more costly than ours in terms of key sizes, encryption time, and decryption time (in particular, encryption takes five exponentiations), but the scheme is still practical.

The notions of indistinguishability and semantic security, and their equivalence under chosen-plaintext attack is due to [44]. The notion of chosen-ciphertext security that we use is due to [68]. Equivalences are further investigated by [9]. Note that the form of chosen-ciphertext security we use is the "strong" form, called CCA2 in [9].

Outline. To specify our scheme in a compact and precise way, we first specify in Section II.B the "syntax" of an asymmetric encryption scheme and what it means for it to be secure. We also specify in Section II.B the syntax of the types of primitives which our asymmetric encryption scheme employs along with their security definitions. The specification of DHIES is then given in Section II.C and its attributes and advantages are discussed in Section II.D.

The security of DHIES relies on variants of the Diffie-Hellman problem, which we introduce in Section II.E. More specifically, we formalize three new

Diffie-Hellman assumptions (though one of them, the hash Diffie-Hellman assumption, is essentially folklore). The assumptions are the *hash* Diffie-Hellman assumption (HDH), the *oracle* Diffie-Hellman assumption (ODH), and the *strong* Diffie-Hellman assumption (SDH). The HDH and ODH assumptions measure the sense in which a hash function H is "independent" of the underlying Diffie-Hellman problem. One often hears intuition asserting that two primitives are independent. Here is one way to define this. The SDH assumption formalizes, in a simple manner, that the "only" way to compute a value g^{uv} from g^v is to choose a value u and compute $(g^v)^u$. The definitions for both ODH and SDH have oracles which play a central role.

Section II.F shows that DHIES is secure against chosen-plaintext attacks. The HDH assumption is what is required to show this. In Section II.G, we show that DHIES is secure against chosen-ciphertext attacks. The ODH assumption is what is required to show this. Of course this means that DHIES is also secure against chosen-plaintext attacks [9] based on the ODH assumption, but in fact we can prove the latter using the HDH assumption (although we do not show it here), a much weaker one.

These two results make additional cryptographic assumptions: in the case of chosen-plaintext attacks, the security of the symmetric encryption scheme; in the case of chosen-ciphertext attacks, the security of the symmetric encryption scheme and the security of the message authentication code. But the particular assumptions made about these primitives are extremely weak.

The ODH assumption is somewhat technical; SDH is rather simpler. In Section II.H, we show that, in the random-oracle model, the SDH assumption implies the ODH assumption. A lower bound for the difficulty of the SDH assumption in the generic model of Shoup [75] is also given in Section II.H. This rules out a large class of efficient attacks.

Following works such as [15, 16], we take a concrete, quantitative approach for all of the results above.

II.B Definitions

II.B.1 Represented groups

DHIES makes use of a finite cyclic group $G = \langle g \rangle$. (This notation indicates that G is generated by the group element g.) We will use multiplicative notation for the group operation. So, for $u \in \mathsf{N}$, g^u denotes the group element of G that results from multiplying u copies of g. Naturally, g^0 names the identity element of G. Note that, if $u \in \mathsf{N}$, then, by Lagrange's theorem, $g^u = g^{u \bmod |G|}$.

Algorithms which operate on G will be given string representations of elements in G. We thus require an injective map $\_ : G \to \{0,1\}^{gLen}$ associated to G, where *gLen* is some number (the length of the representation of group elements). Similarly, when a number $i \in \mathsf{N}$ is an input to, or output of, an algorithm, it must be appropriately encoded, say in binary. We assume all necessary encoding methods are fixed, and do not normally write the $\_$ operators.

Any "reasonable" group supports a variety of computationally feasible group operations. Of particular interest is there being an algorithm $\uparrow$ which takes (the representations of) a group element x and a number i and computes (the representation of) x^i. For clarity, we write this operator in infix, so that $(x) \uparrow (i)$ returns x^i. We will call the tuple $\mathcal{G} = (G, g, \_, \uparrow)$ a *represented group.*

II.B.2 Message Authentication Codes

Let $\mathsf{Message} = \{0,1\}^*$ and let $\mathsf{mKey} = \{0,1\}^{mLen}$ for some number *mLen*. Let $\mathsf{Tag} = \{0,1\}^{tLen}$ for some number *tLen* (a superset of the possible tags). A *message authentication code* is a pair of algorithms $\mathsf{MAC} = (\mathcal{T}, \mathcal{V})$. Algorithm $\mathcal{T}$ (the *MAC generation algorithm*) takes a key $k \in \mathsf{mKey}$ and a message $x \in \mathsf{Message}$ and returns a string $\mathcal{T}(k, x)$. This string is called the *tag*. Algorithm $\mathcal{V}$ (the *MAC verification algorithm*) takes a key $k \in \mathsf{mKey}$, a message $x \in \mathsf{Message}$, and a purported tag $\tau \in \mathsf{Tag}$. It returns a bit $\mathcal{V}(k, x, \tau) \in \{0,1\}$, with 0 indicating that the message was rejected (deemed unauthentic) and 1 indicating that the

message was accepted (deemed authentic). We require that for all $k \in \mathsf{mKey}$ and $x \in \mathsf{Message}$, $\mathcal{V}(k, x, \mathcal{T}(k, x)) = 1$. The first argument of either algorithm may be written as a subscript.

SECURITY. The security of a MAC is defined by an experiment in which we first choose a random key $k \in \mathsf{mKey}$ and then give an adversary F a $\mathcal{T}_k(\cdot)$ oracle, we say that F's output (x^*, τ^*) is *unasked* if τ^* is not the response of the $\mathcal{T}_k(\cdot)$ oracle to an earlier query of x^*. Our definition of MAC security follows.

Definition II.B.1 Let $\mathsf{MAC} = (\mathcal{T}, \mathcal{V})$ be a message authentication scheme and let F be an adversary. Consider the experiment

`experiment` $\mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MAC},F}$

$k \xleftarrow{R} \mathsf{mKey}$

$(x^*, \tau^*) \leftarrow F^{\mathcal{T}_k(\cdot), \mathcal{V}_k(\cdot,\cdot)}$

`if` $\mathcal{V}_k(x^*, \tau^*) = 1$ `and` τ^* was never returned by $\mathcal{T}_k(\cdot)$ in response to query x^*

`then return` 1 `else return` 0

Now define the *suf-cma-advantage* of F as follows:

$$\mathbf{Adv}^{\text{mac}}_{\mathsf{MAC},F} \;=\; \Pr[\,\mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MAC},F} = 1\,]\,.$$

For any t, q_t, μ_t, q_v, and μ_t, we define the *suf-cma-advantage* of MAC as

$$\mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t, q_t, \mu_t, q_v, \mu_v) \;=\; \max_F\{\,\mathbf{Adv}^{\text{mac}}_{\mathsf{MAC},F}\,\}$$

where the maximum is over all F with time-complexity t, making to the tag oracle at most q_t queries the sum of whose lengths is at most μ_t bits and making to the verification oracle at most q_v queries the sum of whose lengths is at most μ_v bits. $\diamondsuit$

We say adversary F has *forged* when, in the experiment above, it outputs a pair (x^*, τ^*) such that $\mathcal{V}_k(x^*, \tau^*) = 1$ and (x^*, τ^*) was not previously obtained via a query to the tag oracle.

This definition is stronger than the usual one as given in [11]. There, one asks that the adversary not be able to produce MACs of new messages. Here

we require additionally that the adversary not be able to generate new MACs of old messages. However, if the MAC generation function is deterministic and verification is done by simply re-computing the MAC (this is typically true) then there is no difference.

CANDIDATES. Candidate algorithms include HMAC [7] or the CBC MAC (but only a version that is correct across messages of arbitrary length).

II.B.3 Symmetric Encryption

Let Message be as before, and let eKey $= \{0,1\}^{eLen}$, for some number *eLen*. Let Ciphertext $= \{0,1\}^*$ (a superset of all possible ciphertexts). Let Coins be a synonym for $\{0,1\}^\infty$ (the set of infinite strings). A *s*ymmetric encryption scheme is a pair of algorithms SYM $= (\mathcal{E}, \mathcal{D})$. Algorithm $\mathcal{E}$ (the *e*ncryption algorithm) takes a key $k \in$ eKey, a plaintext $x \in$ Message, and coins $r \in$ Coins, and returns ciphertext $\mathcal{E}(k, x, r)$. Algorithm $\mathcal{D}$ (the *decryption algorithm*) takes a key $k \in$ eKey and a purported ciphertext $y \in$ Ciphertext, and returns a value $\mathcal{D}(k, y) \in$ Message $\cup \{\text{BAD}\}$. We require that for all $x \in$ Message, $k \in$ Key, and $r \in$ Coins, $\mathcal{D}(k, \mathcal{E}(k, x, r)) = x$. Usually we omit mentioning the coins of $\mathcal{E}$, thinking of $\mathcal{E}$ as a probabilistic algorithm, or thinking of $\mathcal{E}(k, x)$ as the induced probability space. A return value of BAD from $\mathcal{D}$ is intended to indicate that the ciphertext was regarded as "invalid" (it is not the encryption of any plaintext). The first argument of either algorithm may be written as a subscript.

SECURITY. Security of a symmetric encryption scheme is defined as in [8], in turn an adaptation of the notion of polynomial security as given in [44, 61]. We imagine an adversary A that runs in two stages. During either stage the adversary may query an encryption oracle $\mathcal{E}(k, \cdot)$ which, on input x, returns $\mathcal{E}(k, x, r)$ for a randomly chosen r. In the adversary's find stage it endeavors to come up with a pair of equal-length messages, x_0 and x_1, whose encryptions it wants to try to tell apart. It also retains some state information s. In the adversary's guess stage it is given a random ciphertext y for one of the plaintexts x_0, x_1, together with the

saved state s. The adversary "wins" if it correctly identifies which plaintext goes with y. The encryption scheme is "good" if "reasonable" adversaries can't win significantly more than half the time.

Definition II.B.2 [8] Let $\mathsf{SYM} = (\mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. Consider the experiment

$\texttt{experiment}\ \mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{SYM},A}$

$k \xleftarrow{R} \mathsf{eKey}$

$(x_0, x_1, s) \leftarrow A^{\mathcal{E}(k,\cdot)}(\mathsf{find})$

$b \xleftarrow{R} \{0,1\}$

$y \leftarrow \mathcal{E}(k, x_b)$

$\widetilde{b} \leftarrow A^{\mathcal{E}(k,\cdot)}(\mathsf{guess}, y, s)$

$\texttt{if}\ \widetilde{b} = b\ \texttt{then return}\ 1\ \texttt{else return}\ 0$

Now define the *ind-cpa-advantage* of A in the find-and-guess notion as follows:

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},A} \;=\; 2 \cdot \Pr[\,\mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{SYM},A} = 1\,] - 1$$

if A is legitimate, and 0 otherwise. For any t, q, and μ, we define the *ind-cpa-advantage* of SYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t, q, \mu) \;=\; \max_{A}\{\,\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},A}\,\}$$

where the maximum is over all A with time-complexity t, making to the encryption oracle at most q queries the sum of whose lengths is at most μ bits. $\diamondsuit$

It is understood that, above, A must output x_0 and x_1 with $|x_0| = |x_1|$. The multiplication by 2 and subtraction by 1 are just scaling factors, to make a numeric value of 0 correspond to no advantage and a numeric value of 1 correspond to perfect advantage. As a reminder, "time-complexity" is the maximum execution time of the experiment $\mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{SYM},A}$ plus the size of the code for A, all in some fixed RAM model of computation.

Candidates. One candidate algorithms for the symmetric encryption are CBC encryption and Vernam cipher encryption.

II.B.4 Asymmetric Encryption

Let Coins, $\mathsf{Message}$, $\mathsf{Ciphertext}$ be as before and let $\mathsf{PK} \subseteq \{0,1\}^*$ and $\mathsf{SK} \subseteq \{0,1\}^*$ be sets of strings. An *asymmetric encryption scheme* is a three-tuple of algorithms $\mathsf{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$. The *encryption algorithm* $\overline{\mathcal{E}}$ takes a public key $pk \in \mathsf{PK}$, a plaintext $x \in \mathsf{Message}$, and coins $r \in \mathsf{Coins}$, and returns a ciphertext $y = \overline{\mathcal{E}}(k, x, r)$. The decryption algorithm $\overline{\mathcal{D}}$ takes a secret key $sk \in \mathsf{SK}$ and a ciphertext $y \in \mathsf{Ciphertext}$, and returns a plaintext $\overline{\mathcal{D}}(sk, y) \in \mathsf{Message} \cup \{\mathrm{BAD}\}$. The key generation algorithm $\overline{\mathcal{K}}$ takes coins $r \in \mathsf{Coins}$ and returns a pair $(pk, sk) \in \mathsf{PK} \times \mathsf{SK}$. We require that for all (pk, sk) which can be output by $\overline{\mathcal{K}}$, for all $x \in \mathsf{Message}$ and $r \in \mathsf{Coins}$, we have that $\overline{\mathcal{D}}(sk, \overline{\mathcal{E}}(pk, x, r)) = x$. The first argument to $\overline{\mathcal{E}}$ and $\overline{\mathcal{D}}$ may be written as a subscript.

Privacy against chosen-plaintext attack. Our treatment mimics the find-then-guess notion of [8] and follows [44, 61, 42]. The definition is similar to Definition II.B.2, so we state it without further discussion.

Definition II.B.3 Let $\mathsf{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be an asymmetric encryption scheme and let A an adversary. Consider the experiment

$\texttt{experiment}\ \mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{ASYM},A}$

$(sk, pk) \leftarrow \overline{\mathcal{K}}$

$(x_0, x_1, s) \leftarrow A(\mathsf{find}, pk)$

$b \xleftarrow{R} \{0,1\}$

$y \leftarrow \overline{\mathcal{E}}_{pk}(x_b)$

$\widetilde{b} \leftarrow A(\mathsf{guess}, pk, y, s)$

$\texttt{if}\ \widetilde{b} = b\ \texttt{then return}\ 1\ \texttt{else return}\ 0$

Now define the *ind-cpa-advantage* of A in the $\mathsf{find\text{-}and\text{-}guess}$ notion as follows:

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{ASYM},A} \quad = \quad 2 \cdot \Pr[\,\mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{ASYM},A} = 1\,] - 1$$

if A is legitimate, and 0 otherwise. For any t, we define the *ind-cpa-advantage* of

ASYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{ASYM}}(t,c) \;=\; \max_{A}\{\, \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{ASYM},A} \,\}$$

where the maximum is over all A with time-complexity t and whose challenge has length at most c bits. $\diamondsuit$

Privacy against adaptive chosen-ciphertext attack. The definition of chosen-ciphertext security of an asymmetric encryption scheme is very similar to that given in Definition II.B.3. The difference is that here the adversary is given access to a decryption oracle in both stages. So we state it without further discussion.

Definition II.B.4 Let $\mathsf{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be an asymmetric encryption scheme and let A an adversary for its chosen-ciphertext security. Consider the experiment

`experiment` $\mathbf{Exp}^{\text{ind-cca-fg}}_{\mathsf{ASYM},A}$

$(sk, pk) \leftarrow \overline{\mathcal{K}}$

$(x_0, x_1, s) \leftarrow A^{\overline{\mathcal{D}}_{sk}}(\mathsf{find}, pk)$

$b \xleftarrow{R} \{0,1\}$

$y \leftarrow \overline{\mathcal{E}}_{pk}(x_b)$

$\widetilde{b} \leftarrow A^{\overline{\mathcal{D}}_{sk}}(\mathsf{guess}, pk, y, s)$

`if` $\widetilde{b} = b$ `then return` 1 `else return` 0

Now define the *ind-cca-advantage* of A in the find-and-guess notion as follows:

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\mathsf{ASYM},A} \;=\; 2 \cdot \Pr[\, \mathbf{Exp}^{\text{ind-cca-fg}}_{\mathsf{ASYM},A} = 1 \,] - 1$$

if A is legitimate, and 0 otherwise. For any t, we define the *ind-cpa-advantage* of ASYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{ASYM}}(t,c) \;=\; \max_{A}\{\, \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{ASYM},A} \,\}$$

where the maximum is over all A with time-complexity t, making to the decryption oracle at most q queries the sum of whose lengths is at most μ bits. $\diamondsuit$

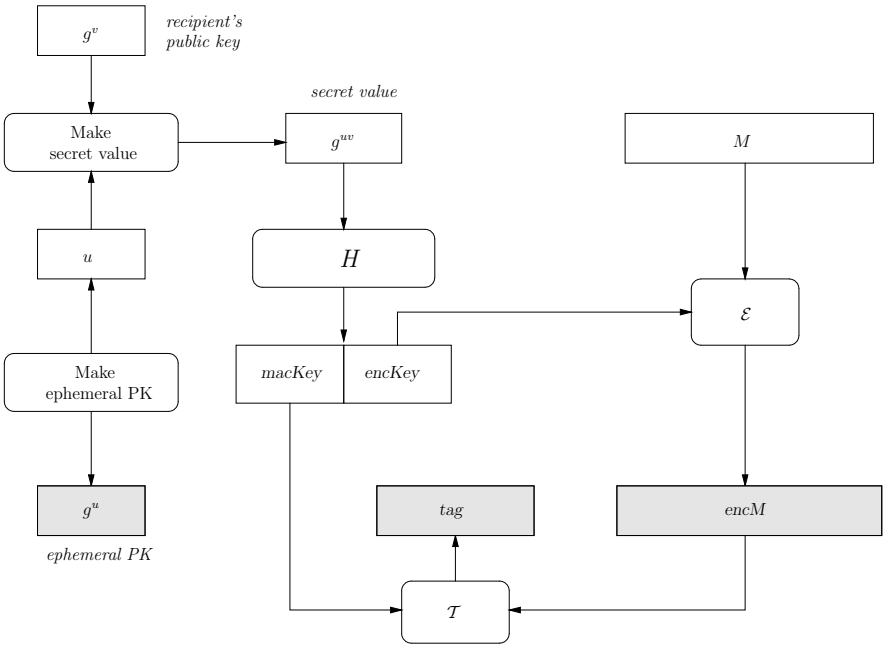

Figure II.1: *Encrypting with the scheme DHIES. We use the symmetric encryption algorithm, $\mathcal{E}$, of* SYM*; the MAC generation algorithm, $\mathcal{T}$, of* MAC*; and a hash function, H. The shaded rectangles comprise the ciphertext.*

II.C The Scheme DHIES

This section recalls the DHIES scheme. Refer to Figure II.1 for a pictorial representation of encryption under DHIES, and Figure II.2 for the formal definition of the algorithm. Let us explain the scheme in reference to those descriptions.

Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, where group elements are represented by strings of *gLen* bits. Let $\mathsf{SYM} = (\mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with key length *eLen*, and let $\mathsf{MAC} = (\mathcal{T}, \mathcal{V})$ be a message authentication code with key length *mLen* and tag length *tLen*. Let $H : \{0,1\}^{gLen} \to \{0,1\}^{mLen+eLen}$ be a function. From these primitives we define the asymmetric encryption scheme DHIES $= (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$. We will write DHIES $[\![\mathcal{G}, \mathsf{SYM}, \mathsf{MAC}, \mathsf{H}]\!]$ whenever we want to explicitly indicate the dependency of DHIES on its associated primitives. The component algorithms of DHIES are defined in Figure II.2.

Each user's public key and secret key is exactly the same as with the

ElGamal scheme: g^v and v, respectively, for a randomly chosen v. (Here we will not bother to distinguish group elements and their bit-string representations.) To send a user an encrypted message we choose a random u and compute an "ephemeral public key," g^u. Including g^u in the ciphertext provides an "implicit" Diffie-Hellman key exchange: the sender and receiver will both be able to compute the "secret value" g^{uv}. We pass g^{uv} to the hash function H and parse the result into two pieces: a MAC key, *macKey*, and an encryption key, *encKey*. We symmetrically encrypt the message we wish to send with the encryption key, and we MAC the resulting ciphertext using the MAC key. The ciphertext consists of the ephemeral public key, the symmetrically encrypted plaintext, and the authentication tag generated by the MAC.

The group G is of prime order. We henceforth assume that $|G|$ is prime. This is extremely important to ensure the security of DHIES or otherwise the scheme could be malleable. The reason stems from the fact that in groups where $|G|$ is not a prime (e.g., Z_p^*), g^{uv} and g^v together might not uniquely determine g^u. That is, there may exist two values u and u' such that $u \neq u'$ but $g^{uv} = g^{u'v}$. As a result, both u and u' would produce two different valid ciphertexts for the same plaintext. Therefore, if one can compute $g^{u'}$, given g^u and g^v, such that $g^{uv} = g^{u'v}$ holds with high probability, then we would break the scheme in the malleability sense. To prevent such attacks in groups not of prime order (e.g., Z_p^*), one should feed g^u to H.

II.D Attributes and Advantages of DHIES

To explain the problem which DHIES solves, and the sense in which it solves this problem, let us back up and provide a bit of background.

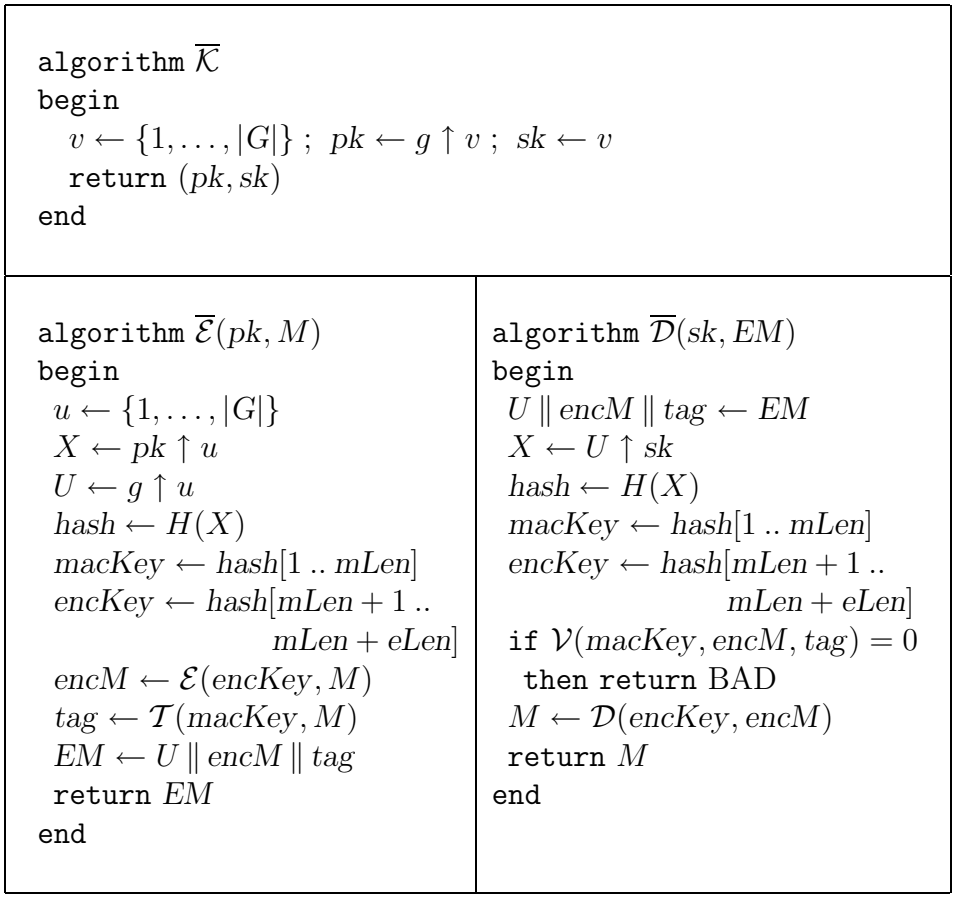

Figure II.2: *The scheme* DHIES $= (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$*, where:* $\mathsf{SYM} = (\mathcal{E}, \mathcal{D})$ *is a symmetric encryption scheme using keys of length eLen;* $\mathsf{MAC} = (\mathcal{T}, \mathcal{V})$ *is a message authentication code with keys of length mLen and tags of length tLen;* $\mathcal{G} = (G, g, \_, \uparrow)$ *is a represented group whose group elements encoded by strings of length gLen; and* $H : \{0,1\}^{gLen} \to \{0,1\}^{eLen+mLen}$.

II.D.1 Encrypting with Diffie-Hellman: The ElGamal Scheme

Let G be a finite cyclic group, say $G = Z_p^*$, the multiplicative group of integers modulo a (large) prime p. We'll denote the group operation of G multiplicatively, so that repeated multiplication is represented by exponentiation. Let g be a generator for G; that is, the elements of G are $\{g^1, g^2, \ldots, g^{|G|}\}$. Fix such a group G and its generator g. All multiplications (or exponentiations, which is just shorthand for repeated multiplication) will be performed in G.

Diffie and Hellman suggested that two parties communicating over a channel subject to (passive) eavesdropping could come to share a secret key as follows [32]. The first party chooses a random number $u \in \{1, \ldots, |G|\}$ and sends g^u to the second party. The second party chooses a random number $v \in \{1, \ldots, |G|\}$ and sends g^v to the first party. The shared key is declared to be g^{uv}, which the first party can calculate as $(g^v)^u$ and the second party can calculate at $(g^u)^v$.

Roughly said, the *Diffie-Hellman assumption* for G asserts that an adversary who sees g^u and g^v (for a random u, v) cannot compute g^{uv}.

ElGamal [35] explained how to adapt the above to give a public key encryption method. The intended receiver of an encrypted message has a public key which specifies g^v (where v was chosen randomly from $\{1, \ldots, |G|\}$). The sender wants to send to that receiver a ciphertext C which is the encryption of a message M. We assume $M \in G$. The sender computes C by choosing a random u (again in $\{1, \ldots, |G|\}$) and transmitting $C = (g^u, M \cdot g^{uv})$. Knowing v, the receiver can compute $g^{uv} = (g^u)^v$ from C and then multiply $M \cdot g^{uv}$ by the inverse of g^{uv} to recover M.

II.D.2 Deficiencies of ElGamal Encryption

We highlight a number of issues arising from the encryption method we have just described.

1. *Limited message space.* First there was the assumption that $M \in G$. Messages are naturally regarded as bit strings, not group elements. Often there will be a natural embedding of *some* bit strings into group elements, but that may fall short of all potential messages.

2. *May not provide good privacy.* As Goldwasser and Micali explain and formalize in [44], a good encryption scheme should do more than make it infeasible for an adversary to decrypt: the scheme should conceal from an adversary mounting a passive attack "any" information about the plaintext. For example, it should not be possible to determine even one bit of the plaintext given the ciphertext.

This property has been defined in several ways which have been shown to be equivalent [44], including a definitions known as "indistinguishability" and one known as "semantic security."

Even in groups for which one anticipates using ElGamal encryption, the ElGamal encryption does not achieve semantic security. For example, when the scheme is implemented in the group $G = \mathsf{Z}_p^*$, there are attacks showing that some information about the plaintext can be determined from the ciphertext. See below the description of one such attack.

It is possible to guarantee the semantic security of ElGamal encryption if it is done in special groups, and if we make a stronger assumption about the Diffie-Hellman problem. Specifically, the order of the group should be prime (note the order of Z_p^* is $p-1$ which is not prime) and we make the *decisional Diffie-Hellman* assumption, which says that it is infeasible to distinguish the following two distributions: (g^u, g^v, g^{uv}), for a random u and v, and (g^u, g^v, g^z), for a random u,v, and z. This is a very strong assumption.

It would be preferable to have a scheme which worked in any group where the Diffie-Hellman problem is hard, and one which was guaranteed to achieve semantic security under a weaker number-theoretic assumption.

3. *We want more than basic privacy.* For an encryption scheme to be a maximally useful tool in the design of higher-level protocols it should actually do *m*ore than shield information about the plaintext in the presence of a passive attack. Stronger goals include non-malleability [31] and chosen-ciphertext security [66, 68]. Informally, non-malleability means that an adversary cannot mutate one ciphertext into a related one. Chosen-ciphertext security means that an adversary cannot break an encryption scheme even if it can cause some ciphertexts to be decrypted. ElGamal encryption achieves neither of these "beyond semantic security" goals: it is easy to see that the scheme is malleable and also insecure under a chosen-ciphertext attack. (See below).

We are finding that uses of encryption in cryptographic practice relies

more and more on the scheme meeting these "beyond semantic security" goals. For example, the designers of SET (Secure Electronic Transactions) mandated the use of an encryption scheme which achieves more than semantic security. This was necessary, in the sense that the SET protocols would be *w*rong if instantiated by a primitive which achieves *o*nly semantic security, and to design SET-like protocols using a primitive which achieves only semantic security would seem to yield more complicated protocols. As a second example, Bleichenbacher [20] has shown that encryption under RSA PKCS #1 v1.5 [67] is vulnerable to chosen-ciphertext attack, and he goes on to demonstrate how this leads to an attack on SSL 3.0. Because schemes which achieve "only" semantic security are so easily misused by protocol designers, we believe it is highly desirable that standardized schemes achieve "beyond semantic security" goals, particularly non-malleability and chosen-ciphertext security.

Attacks on the ElGamal Scheme

ElGamal encryption fails to achieve strong notions of security, such as non-malleability and chosen-ciphertext security, in any represented group. In fact, it does not even achieves semantic security in some groups, such as Z_p^*. To support these claims, we here provide the reader with examples of attacks on the ElGamal scheme.

The first of these attacks against the ElGamal scheme shows that it is not semantically secure when Z_p^* is the underlying group of $\mathcal{G}$, p is a prime, and g is a generator. The attack is based on the fact that we can check whether a number $x \in \mathsf{Z}_p^*$ is a square or not in polynomial time by computing the value $x^{(p-1)/2} \bmod p$, which is 1 if x is a quadratic residue mod p and -1, otherwise. In the find stage, we choose two messages in Z_p^*, one which is a square and one which is not. In the guess stage, we first check whether g^u and g^v are square. We know that g^{uv} is a non-square if and only if both g^u and g^v are non-square. Then, knowing this, we can tell which message was encrypted by checking whether the encrypted message

$M \cdot g^{uv}$ is a square or not. That is, if g^{uv} is a square, then $M \cdot g^{uv}$ is a square if and only if M is a square. If g^{uv} is a non-square, then $M \cdot g^{uv}$ is a square if and only if M is a non-square.

In order to provide a malleability attack against the ElGamal scheme, we can see that, given a ciphertext $EM = (g^u, encM)$ where $encM = M \cdot g^{uv}$, we can easily produce a valid ciphertext $\widetilde{EM}$ by just modifying the second part of EM. That is, if we multiply $encM$ by some value g^k $(k \neq 0)$ to obtain $\widetilde{encM}$, then the resulting ciphertext $\widetilde{EM} = (g^u, encM)$ will be an encryption for a message $\widetilde{M} = M \cdot g^k$ because the value of g^{uv} does not change in this case. Note that this is not dependent on which group G is being used.

To provide a chosen-ciphertext attack against the ElGamal scheme, we can show that we can obtain the plaintext for any given ciphertext. Let $EM = (g^u, encM)$ be the challenge ciphertext. Let $\widetilde{encM}$ be a point in G such that $\widetilde{encM} \neq encM$ and let $\widetilde{M}$ be the decryption of $\widetilde{EM} = (g^u, \widetilde{encM})$. As we know that $\widetilde{M} = \widetilde{encM}/g^{uv}$, we can compute g^{uv} and then $encM/g^{uv}$, which is the decryption of EM.

II.D.3 Overcoming Deficiencies in ElGamal Encryption: DHIES

The scheme we have presented, DHIES, does Diffie-Hellman based encryption in a way which overcomes the limitations enumerated above, but without significant increase in cost compared to ElGamal. Key characteristics and advantages of DHIES include the following.

1. *Basic privacy — Proven in the sense of provable security.* Roughly said, to achieve semantic security we assume the existence of a function $H : G \rightarrow \{0,1\}^*$ such that $\langle g^u, g^v, H(g^{uv})\rangle$ looks like a pair of random group elements together with a random string. For non-trivial functions H this assumption —that H is hard-core for the Diffie-Hellman problem on G— would seem to be weaker than decisional Diffie-Hellman. We prove that under this assumption, our scheme achieves semantic security. For reasonable choices of H, this assumption would

seem to hold for any group one would imagine using, not just particular groups.

2. *Beyond basic privacy: non-malleability and chosen-ciphertext security — proven in the sense of provable security.* We prove that our scheme is secure against adaptive chosen-ciphertext attacks. This is proved under an assumption called the Oracle Diffie-Hellman assumption, and assuming the underlying MAC and encryption schemes are secure. It is shown in [9, 34] that security under adaptive chosen-ciphertext attack implies non-malleability, so that property is achieved automatically.

3. *No random oracles.* The proofs here do not appeal to the random oracle model. They are all in the standard model. This addresses concerns that have been raised about this model [26].

4. *Efficiency.* The efficiency of ElGamal encryption is preserved: the cost of encryption is essentially the same as with ElGamal encryption: two exponentiations to encrypt, one to decrypt. For encryption, both of these exponentiations can be *o*ff-line, meaning that they can be done even before the message M is known. The length of ciphertexts and the public key is the same as in ElGamal.

5. *Versatile instantiation — The group.* We allow considerable versatility in instantiating DHIES. First, the group G in which we perform our operations can be essentially any group in which our version of the Diffie-Hellman assumption is reasonable. It could be Z_p^*, or a subgroup of Z_n^*, or an elliptic curve group (in which case the group operation is usually written additively, so what we have been denoting g^u would be written multiplicatively, as ug). Our proofs assume no algebraic structure for G beyond its being a finite cyclic group.

6. *Versatile instantiation — Ancillary primitives.* Cryptography beyond the group operations is performed using generic primitives. We employ primitives for symmetric encryption, message authentication, and hashing. For achieving semantic security, the underlying symmetric encryption and hashing schemes must meet weak, formalized assumptions. For achieving non-malleability and chosen-ciphertext security the encryption scheme and message authentication code must

meet weak, formalized assumptions, while the hash function is modeled by a public random oracle.

7. *Arbitrary message space.* Finally, messages to be encrypted are arbitrary bit strings; messages are not restricted in length or content.

II.E Diffie-Hellman Assumptions

This section specifies five versions of the Diffie-Hellman assumption. The first two are standard (included here only for completeness); the next one is straightforward/folklore; and the last assumptions are new.

Computational Diffie-Hellman assumption: CDH. We refer to the "standard" Diffie-Hellman assumption as the *computational Diffie-Hellman* assumption, CDH. It states that given g^u and g^v, where u, v were drawn at random from $\{1, \ldots, |G|\}$, it is hard to compute g^{uv}. Under the computational Diffie-Hellman assumption it might well be possible for the adversary to compute something interesting about g^{uv} given g^u and g^v; for example, the adversary might be able to compute the most significant bit, or even half of the bits. This makes the assumption too weak to directly use in typical applications. For example, the ElGamal scheme is not semantically secure given only this assumption.

Definition II.E.1 [Computational Diffie-Hellman: CDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiment

`experiment` $\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},A}$

$u \xleftarrow{R} \{1, \ldots, |G|\}$; $U \leftarrow g^u$

$v \xleftarrow{R} \{1, \ldots, |G|\}$; $V \leftarrow g^v$

$Z \leftarrow A(U, V)$

`if` $Z = g^{uv}$ `then` $b \leftarrow 1$ `else` $b \leftarrow 0$

`return` b

Now define the *advantage* of A in violating the computational Diffie-Hellman as-

sumption as

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G},A} \;=\; \Pr[\,\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

Decisional Diffie-Hellman assumption: DDH. A stronger assumption that has been gaining popularity is the *decisional Diffie-Hellman* assumption, DDH. (For a nice discussion, see Boneh's survey [24].) It states, roughly, that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are drawn at random from $\{1, \ldots, |G|\}$. This assumption can only hold in a group G whose order does not contain small prime factors (e.g., subgroup of order q of Z_p^* for large primes p and q). In such groups the assumption suffices to prove the semantic security of the ElGamal scheme.

Definition II.E.2 [Decisional Diffie-Hellman: DDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiments

$\texttt{experiment}\ \mathbf{Exp}^{\text{ddh-real}}_{\mathcal{G},A}$	$\texttt{experiment}\ \mathbf{Exp}^{\text{ddh-rand}}_{\mathcal{G},A}$
$u \xleftarrow{R} \{1, \ldots, \|G\|\}\ ;\ U \leftarrow g^u$	$u \xleftarrow{R} \{1, \ldots, \|G\|\}\ ;\ U \leftarrow g^u$
$v \xleftarrow{R} \{1, \ldots, \|G\|\}\ ;\ V \leftarrow g^v$	$v \xleftarrow{R} \{1, \ldots, \|G\|\}\ ;\ V \leftarrow g^v$
$Z \leftarrow g^{uv}$	$z \xleftarrow{R} \{1, \ldots, \|G\|\}\ ;\ Z \leftarrow g^z$
$b \leftarrow A(U, V, Z)$	$b \leftarrow A(U, V, Z)$
$\texttt{return}\ b$	$\texttt{return}\ b$

Now define the *advantage* of A in violating the decisional Diffie-Hellman assumption as

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},A} \;=\; \Pr[\,\mathbf{Exp}^{\text{ddh-real}}_{\mathcal{G},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ddh-rand}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

The assumption we make to prove security for DHIES under chosen-plaintext attack is weaker than DDH but stronger than CDH. It is called the *hash Diffie-Hellman* assumption, HDH. To prove the security of DHIES under chosen-ciphertext attacks, we will make stronger versions of the Hash Diffie-Hellman assumptions which say the assumption is true even when the adversary has additional power in the form of oracles giving certain kinds of information about other,

independent Diffie-Hellman keys. The precise formulation of all three of our assumptions is below, and they are followed by a discussion on the choice of hash functions suitable for these assumptions.

HASH DIFFIE-HELLMAN ASSUMPTION: HDH. As indicated above, semantic security of a Diffie-Hellman-based scheme requires that we be able to get some number of "hard-core" bits from the Diffie-Hellman key, namely key derived bits that cannot be distinguished from random bits. Our assumption is that applying a suitable hash function H to g^{uv} will yield such bits. The assumption we make, called the *Hash Diffie-Hellman* assumption, HDH, is a "composite" one—it concerns the interaction between a hash function H and the group operations in G. Here is the definition.

Definition II.E.3 [Hash Diffie-Hellman: HDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let $hLen$ be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

$\texttt{experiment}\ \mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\mathsf{H},A}$	$\texttt{experiment}\ \mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\mathsf{H},A}$
$u \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ U \leftarrow g^u$	$u \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ U \leftarrow g^u$
$v \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ V \leftarrow g^v$	$v \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ V \leftarrow g^v$
$Z \leftarrow H(g^{uv})$	$Z \xleftarrow{R} \{0,1\}^{hLen}$
$b \leftarrow A(U,V,Z)$	$b \leftarrow A(U,V,Z)$
$\texttt{return}\ b$	$\texttt{return}\ b$

Now define the *advantage* of A in violating the hash Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H},A} \;=\; \Pr[\,\mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\mathsf{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\mathsf{H},A} = 1\,]\,. \qquad \diamondsuit$$

The decisional Diffie-Hellman assumption says that g^{uv} looks like a random group element, even if you know g^u and g^v. The hash Diffie-Hellman assumption says that $H(g^{uv})$ looks like a random string, even if you know g^u and g^v. So if you set H to be the identity function you almost recover the decisional Diffie-Hellman assumption (the difference being that in one case you get a random group

element and in the other you get a random string). When H is a cryptographic hash function, like SHA-1 [72], the hash Diffie-Hellman assumption would seem to be a much weaker assumption than the decisional Diffie-Hellman assumption.

We now move on to some more novel assumptions.

Oracle Diffie-Hellman assumption: ODH. Suppose we provide an adversary A with g^v and an oracle $\mathcal{H}_v$, which computes the function $\mathcal{H}_v(X) = X^v$. Think of $v \in \{1, \ldots, |G|\}$ as having been chosen at random. Now if we give the adversary g^u (where $u \in \{1, \ldots, |G|\}$ is chosen at random) then the oracle will certainly enable the adversary to compute g^{uv}: the adversary need only ask the query g^u and she gets back $\mathcal{H}_v(g^u) = g^{uv}$. Even if we *forbid* the adversary from asking g^u, still she can exploit the self-reducibility of the discrete log to find the value of g^{uv}. For example, the adversary could compute $\mathcal{H}_v(gg^u) = g^{uv}g^v$ and divide this by $\mathcal{H}_v(1) = g^v$.

But what if instead we give the adversary an oracle $\mathcal{H}_v$ which computes $\mathcal{H}_v(X) = H(X^v)$, for H a cryptographic hash function such as SHA-1? Suppose the adversary's goal is to compute $H(g^{uv})$, where g^u and g^v are provided to the adversary. Now, as long as the oracle $\mathcal{H}_v$ can not be queried at g^u, the oracle would seem to be useless. We formalize this as follows.

Definition II.E.4 [Oracle Diffie-Hellman: ODH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let *hLen* be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

$\texttt{experiment } \mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\mathsf{H},A}$	$\texttt{experiment } \mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\mathsf{H},A}$
$u \xleftarrow{R} \{1, \ldots, \lvert G\rvert\}\ ;\ U \leftarrow g^u$	$u \xleftarrow{R} \{1, \ldots, \lvert G\rvert\}\ ;\ U \leftarrow g^u$
$v \xleftarrow{R} \{1, \ldots, \lvert G\rvert\}\ ;\ V \leftarrow g^v$	$v \xleftarrow{R} \{1, \ldots, \lvert G\rvert\}\ ;\ V \leftarrow g^v$
$W \leftarrow H(g^{uv})$	$W \xleftarrow{R} \{0,1\}^{hLen}$
$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$	$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$
$b \leftarrow A^{\mathcal{H}_v(\cdot)}(U, V, W)$	$b \leftarrow A^{\mathcal{H}_v(\cdot)}(U, V, W)$
$\texttt{return } b$	$\texttt{return } b$

Now define the *advantage* of A in violating the oracle Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H},A} \;=\; \Pr[\,\mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\mathsf{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\mathsf{H},A} = 1\,]\,.$$

Here A is not allowed to call its oracle on g^u. $\diamondsuit$

We emphasize that the adversary is allowed to make oracle queries that depend on the target g^u, with the sole restriction of not being allowed to query g^u itself.

Strong Diffie-Hellman assumption: SDH. Suppose A is an algorithm which, given g^v, outputs a pair of strings (g^u, g^{uv}), for some $u \in \{1, \ldots, |G|\}$. One way for A to find such a pair is to pick some value u and then compute g^u and g^{uv}. Indeed, we expect this to be the "only" way A can compute such a pair of values. We capture this idea as follows.

Given a represented group $\mathcal{G} = (G, g, \_, \uparrow)$ and a number v, let $\mathcal{O}_v$ be an oracle, called a *restricted DDH oracle*, which behaves as follows:

$$\mathcal{O}_v(U, X) \;=\; \begin{cases} 1 & \text{if } X = U^v \\ 0 & \text{otherwise} \end{cases}$$

That is, the oracle tells whether the second argument equals the first argument raised to v-th power. This oracle can be seen as a restricted form of a DDH oracle for which we fix one of its arguments as being g^v. Our next definition speaks to the uselessness of having a restricted DDH oracle.

Definition II.E.5 [Strong Diffie-Hellman: SDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiment

$\texttt{experiment}\ \mathbf{Exp}^{\text{sdh}}_{\mathcal{G},A}$

$u \xleftarrow{R} \{1,\ldots,|G|\}\ ;\ U \leftarrow g^u$

$v \xleftarrow{R} \{1,\ldots,|G|\}\ ;\ V \leftarrow g^v$

$\mathcal{O}_v(U,X) \stackrel{\text{def}}{=} (X = U^v)$

$Z \leftarrow A^{\mathcal{O}_v(\cdot,\cdot)}(U,V)$

$\texttt{if}\ Z = g^{uv}\ \texttt{then}\ b \leftarrow 1\ \texttt{else}\ b \leftarrow 0$

$\texttt{return}\ b$

Now define the *advantage* of A in violating the strong Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{sdh}}_{\mathcal{G},A} \;=\; \Pr[\,\mathbf{Exp}^{\text{sdh}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

The intuition is that the restricted DDH oracle is useless because the adversary already "knows" the answer to almost any query it will ask.

Similar intuition was captured in [47] by saying that for every non-uniform probabilistic polynomial-time algorithm A that, on input g^v, outputs (g^u, g^{uv}), there exists a non-uniform probabilistic polynomial-time algorithm S (the "extractor") that not only outputs (g^u, g^{uv}), but also u. Our approach avoids the complexity of a simulator-based formulation. We emphasize that our oracle does not return a value u (the discrete log of its first argument) but only a bit indicating whether a given pair has the right form.

Resource measures. We have defined several different senses of adversarial advantage. For each notion xxx we overload the notation and define

$$\mathbf{Adv}^{\text{xxx}}_{\Pi}(R) \;=\; \max_{A}\{\,\mathbf{Adv}^{\text{xxx}}_{\Pi,A}\,\}$$

where R is a resource measure and the maximum is taken over all adversaries that use resources at most R. The resources of interest in our case are time-complexity (denoted by t) and, when appropriate, number of queries (denoted by q). Any other resources of importance will be mentioned when the corresponding notion is described. Here and throughout this dissertation "time-complexity" is understood

to mean the maximum of the execution times of the experiments defining the advantage of adversary A plus the size of the code for A, all in some fixed RAM model of computation. (Note that the execution time refers to that of the entire experiment, not just the execution time of the adversary.)

We comment that we are considering the complexity of adversaries who try to attack a specific represented group $\mathcal{G}$. Such an adversary may depend on $\mathcal{G}$, so explicitly providing a description of $\mathcal{G}$ to A is unnecessary.

Choice of hash function. Now that we understand how we want the hash function to interact with the group, we can consider various choices for the hash function H.

Our suggested choice is to appropriately derive H from some cryptographic hash function like SHA-1 [72]. (The precise manner in which H is derived from SHA-1 is important and should be discussed.) A primary reason we prefer a cryptographic function is that one-wayness of H appears important to the oracle Diffie-Hellman assumption: it should be hard to recover g^{uv} from $H(g^{uv})$, since otherwise the self-reducibility-based attack we discussed above can be mounted.

Let us back up a bit and try to see what requirements the different assumptions impose on the choice of H. Suppose first we are interested only in semantic security, namely we need just the HDH assumption. There is no known choice of H for which one can prove the hard-coreness under the CDH assumption. Under the DDH assumption, however, things get much easier, since this assumption already says that the Diffie-Hellman key is indistinguishable from a random group element: the only remaining problem is to go from a random group element to a random string of appropriate length. In some groups this can be done quite easily by simple truncation of the key. Alternatively, Naor and Reingold [65] show that application of a function H chosen at random from a family of universal hash functions will suffice. Zheng and Seberry [83] had earlier suggested the application of a universal hash function to the Diffie-Hellman key as a heuristic under the computational Diffie-Hellman assumption. The result of [65] says that under the

stronger DDH assumption this heuristic is valid. Note this function can be chosen at random once and for all and included in the public key. In [83], the function is chosen anew for each encryption and included in the ciphertext, which increases the size of the ciphertext.

However, the use of truncation or universal hash functions appears more dangerous when we come to consider the stronger oracle Diffie-Hellman assumption above. In particular, the result of Boneh and Venkatesan [25] showing that computing the most significant bits of Diffie-Hellman keys is as hard as computing the key itself can be turned on its head to give an algorithm to attack the ODH assumption. Namely, their results show that for some simple choices of functions H, an adversary can use the HDH oracle $\mathcal{H}_v$ defined above to solve the Diffie-Hellman problem. These attacks do not appear to work when a one-way cryptographic hash function is used, which is why we recommend this choice. We do not know whether these attacks rule out all choices of universal hash families, but they do seem to rule out some particular ones.

II.F Security against Chosen-Plaintext Attack

We show that DHIES $[\![\mathcal{G}, \mathsf{SYM}, \mathsf{MAC}, \mathsf{H}]\!]$ meets the notion of indistinguishability under a chosen-plaintext attack, as defined in Definition II.B.3.

Theorem II.F.1 Let $\mathcal{G}$ be a represented group, let SYM be a symmetric encryption scheme, let MAC be a message authentication scheme, and let H be a function. Let DHIES be the asymmetric key encryption scheme associated to these primitives, as defined in Section II.C. Then, for any numbers t and c,

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES}}(t, c) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},H}(t) + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t, 0, 0) \;.$$

Idea of proof. The assumption is that the symmetric encryption scheme SYM is secure and H is hard-core for the Diffie-Hellman problem in the

underlying group. (The assumption that MAC is secure is not needed to ensure semantic security.) The proof considers an adversary A who defeats the semantic security of the scheme. Let g^v be the recipient public key and let $y = U \,\|\, encM \,\|\, tag$ be the challenge ciphertext that this adversary gets in its guess stage. We consider two cases depending on whether the output of H "looks random".

- *Case 1 — The output of H looks random.* In this case, we present an adversary B that breaks the encryption scheme SYM.
- *Case 2 — The output of H does not look random.* In this case, we present an algorithm C that breaks the hard-coreness of H on $\mathcal{G}$.

The formal proof, given below, does not actually consider separate cases, but the underlying intuition is the same. Given A, we construct B and C and then relate A's advantage to that of B and C.

Proof: Let A be an adversary attacking DHIES in the sense of semantic security. Assume it has time-complexity at most t. We construct an adversary B attacking SYM and an adversary C attacking H being hard-core for $\mathcal{G}$, and then upper bound the advantage of A in terms of the advantages of these adversaries.

Algorithm B. Figure II.3 describes algorithm B. Recall from Definition II.B.2 that B has access to an oracle for encryption, and runs in two stages. Notice that B never invokes its encryption oracle $\mathcal{E}$. Moreover, the running time of $\mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B}$ is at most t.

Algorithm C. Figure II.4 depicts the behavior of algorithm C. C is given as input U, V, W, where $U = g^u$ and $V = g^v$ for random u, v, and W is either $H(g^{uv})$ or a random string. C outputs at the end a bit indicating its guess as to which of these cases occurs. Notice that the time-complexity of C is at most t.

Analysis. When $W = H(g^{uv})$ we notice that C is running A as the latter would be run in its attack on the semantic security of DHIES. From the definition of

```
algorithm B^{E(·)}(find)
begin
    v <-R {1,...,|G|}
    pk ← g^v
    (x_0, x_1, s) ← A(find, pk)
    s̃ ← (x_0, x_1, s, pk)
    return (x_0, x_1, s̃)
end
```

```
algorithm B^{E(·)}(guess, ỹ, s̃)
begin
    parse s̃ as (x_0, x_1, s, pk)
    u <-R {1,...,|G|} ; U ← g^u
    macKey <-R {0,1}^{mLen}
    tag ← T_{macKey}(ỹ)
    y ← U ‖ ỹ ‖ tag
    b ← A(guess, pk, s, y)
    return b
end
```

Figure II.3: *Algorithm B for attacking the security of* SYM.

```
algorithm C(U, V, W)
begin
    macKey ← W[1...mLen] ;
    encKey ← W[mLen + 1...mLen + eLen]
    pk ← V
    (x_0, x_1, s) ← A(find, pk)
    b̃ ← {0,1} ;   encM ← E_{encKey}(x_{b̃})
    tag ← T_{macKey}(encM)
    y ← U ‖ encM ‖ tag
    b ← A(guess, pk, s, y)
    if b = b̃ then return 1 else return 0
end
```

Figure II.4: *Algorithm C for attacking the hard-coreness of H on $\mathcal{G}$.*

$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}$, we have that

$$\Pr[\,\mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\mathsf{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2}\;.$$

On the other hand, when W is a random string, we notice that C runs A in the same way as B does, and hence

$$\Pr[\,\mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\mathsf{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B}}{2}\;.$$

Subtracting gives us

$$\begin{aligned}\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H},C} &= \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2} - \frac{1}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B}}{2} \\ &= \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B}}{2} \ ;\end{aligned}$$

whence

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A} \ = \ 2 \cdot \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H},C} + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B} \ .$$

Since the time-complexity of algorithm C is at most t, we conclude that $\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H},C} \leq \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H}}(t)$. Moreover, since B makes 0 encryption queries and has time-complexity at most t, we also have $\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B} \leq \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0)$. Thus from the above we have

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A} \ \leq \ 2 \cdot \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathsf{H}}(t) + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0) \ .$$

But A was an arbitrary adversary subject to the constraint that it had time-complexity at most t and the length of its challenge ciphertext is at most c. The theorem follows. ∎

II.G Security against Chosen-Ciphertext Attack

We show that DHIES $[\![\mathcal{G}, \mathsf{SYM}, \mathsf{MAC}, \mathsf{H}]\!]$ meets the notion of indistinguishability under an adaptive chosen-ciphertext attack, as in Definition II.B.4.

Theorem II.G.1 Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let SYM be a symmetric encryption scheme, and let MAC be a message authentication scheme. Let DHIES be the asymmetric encryption scheme associated to these primitives as defined in Section II.C. Then for any numbers t, q, μ, and c,

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES}}(t,q,\mu,c) \ \leq \ & \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0) + 2 \cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H}}(t,q) + \\ & 2 \cdot \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu) \ .\end{aligned}$$

Idea of Proof. The assumption is that both symmetric encryption scheme SYM and the message authentication scheme MAC are secure and H is

a hard-core for the Diffie-Hellman problem on $\mathcal{G}$ under adaptive Diffie-Hellman attack. The proof considers an adversary A who defeats the adaptive chosen-ciphertext security of the scheme. Let g^v be the recipient public key; let $y = U \,\|\, encM \,\|\, tag$ be the challenge ciphertext that algorithm A gets in its guess stage. Let us call a Type 1 query a ciphertext of the form $U \,\|\, \widetilde{encM} \,\|\, \widetilde{tag}$. A Type 2 query have the form $\widetilde{U} \,\|\, \widetilde{encM} \,\|\, \widetilde{tag}$ with $\widetilde{U} \neq U$. We consider three cases depending on whether the output of H looks random and on whether there was a Type 1 query $\widetilde{y}$ to the decryption oracle $\overline{\mathcal{D}}_{sk}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$.

- *Case 1 — The output of H does not look random.* In this case we present an algorithm C that breaks the hard-coreness of H on $\mathcal{G}$ under adaptive Diffie-Hellman attack.

- *Case 2 — The output of H looks random and there was a* Type 1 *query $\widetilde{y}$ to $\overline{\mathcal{D}}_{sk}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$.* In this case we present an adversary F which breaks the message authentication scheme MAC.

- *Case 3 — The output of H looks random and there was not a* Type 1 *query $\widetilde{y}$ to $\overline{\mathcal{D}}_{sk}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$.* In this case we present an adversary B which breaks the encryption scheme SYM.

Proof: Let A be an adversary attacking DHIES in the sense of adaptive chosen-ciphertext security. Assume it has running time at most t, makes at most q queries to its decryption oracle. We construct an adversary B attacking SYM, an adversary C attacking H being a hard-core for $\mathcal{G}$ under non-adaptive Diffie-Hellman attack, and an adversary F for the message authentication scheme MAC and then upper bound the advantage of A in terms of the advantages of these adversaries.

Algorithm B. Figure II.5 describes algorithm B. Recall from Definition II.B.2 that B has access to an oracle for encryption and runs in two stages. Since the time-complexity t of A accounts for the time taken by decryption queries as well as the time to generate pk and the challenge ciphertext y, the time-complexity of

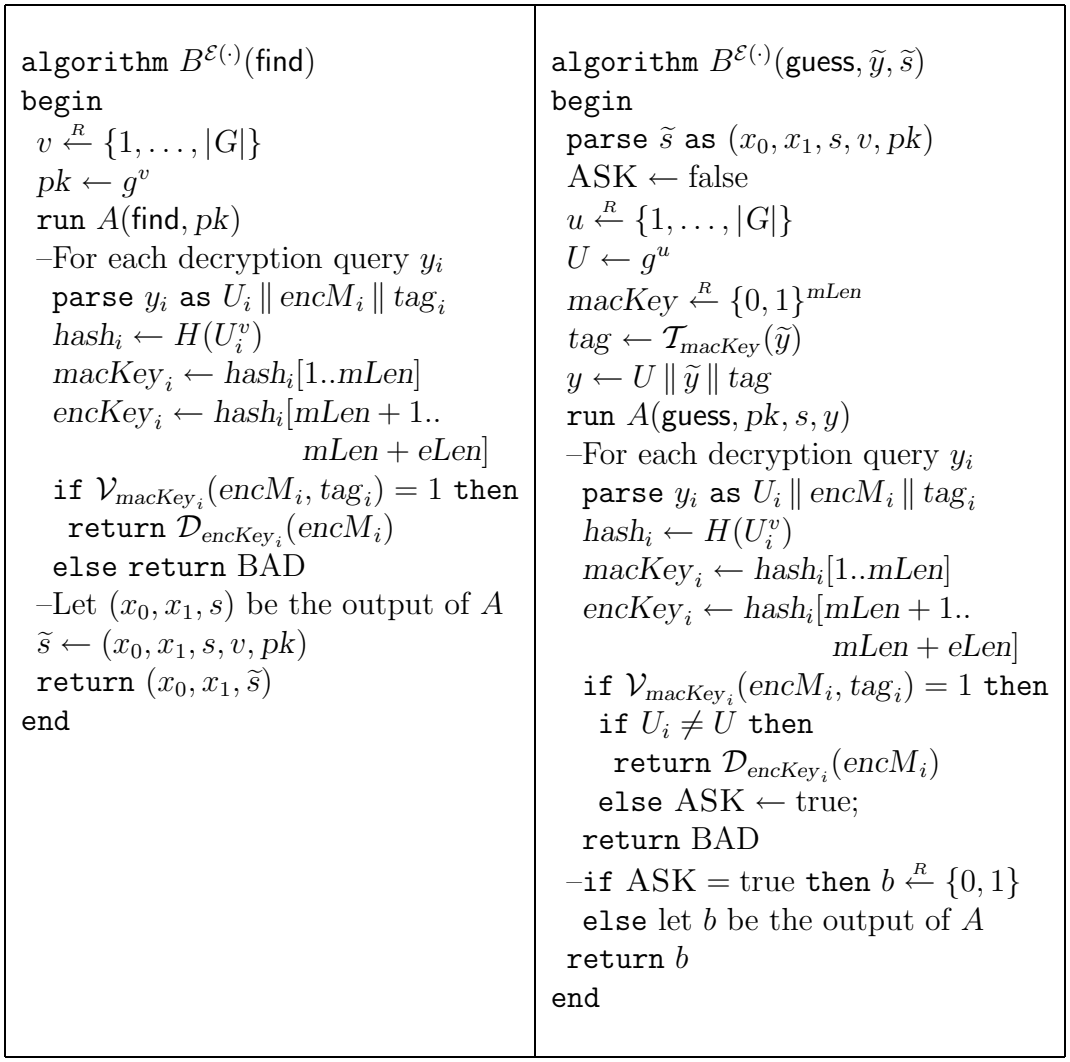

Figure II.5: *Algorithm B for attacking the security of* SYM.

B is at most that of A (i.e., t).

ALGORITHM C. Figure II.6 defines the behavior of algorithm C. C is given as input U, V, W, where $U = g^u$ and $V = g^v$ for random u and v, respectively, and W is either $H(g^{uv})$ or a random string. Recall from Definition II.E.4 that C is also given access to a $\mathcal{H}_v$-oracle. At the end, C outputs a bit indicating its guess as to which of these cases occurs.

Notice that, since the time-complexity of A accounts for the time taken by decryption queries as well as the time to compute the challenge ciphertext, the time-

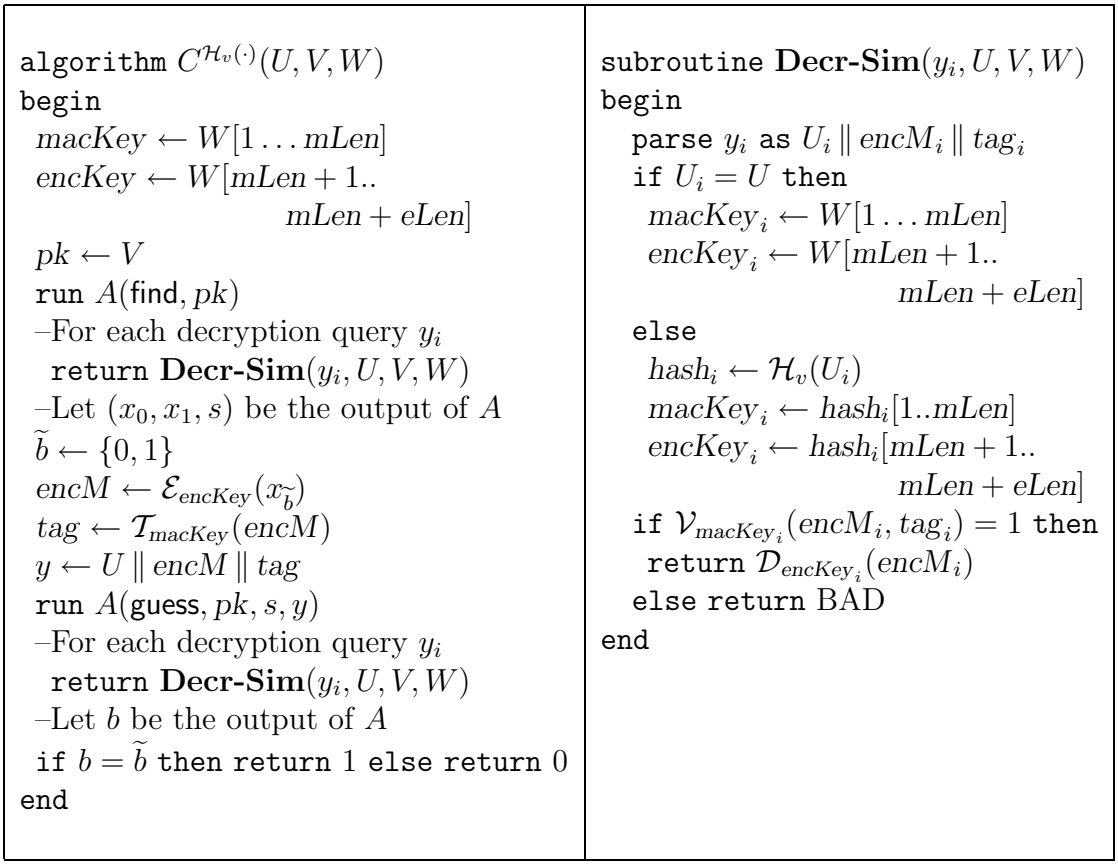

Figure II.6: *Algorithm C for attacking the hard-coreness of H on $\mathcal{G}$ under adaptive Diffie-Hellman attack.*

complexity of C is at most t.

Algorithm F. Figure II.7 describes algorithm F. Recall from Definition II.B.1 that F has access to two oracles: a tag-generation oracle $\mathcal{T}$ and a verification oracle $\mathcal{V}$. It outputs a pair message-tag, a possible forgery. Notice that, since the time complexity of A accounts for the time taken by decryption queries and for the time to generate the secret-public key pair sk, pk) and the challenge ciphertext y, F's time-complexity is at most t.

Analysis. As in the proof of non-adaptive chosen-ciphertext security, our goal is to upper bound the success probability of adversary A in terms of the success probabilities of adversaries B for the symmetric encryption scheme, C for the

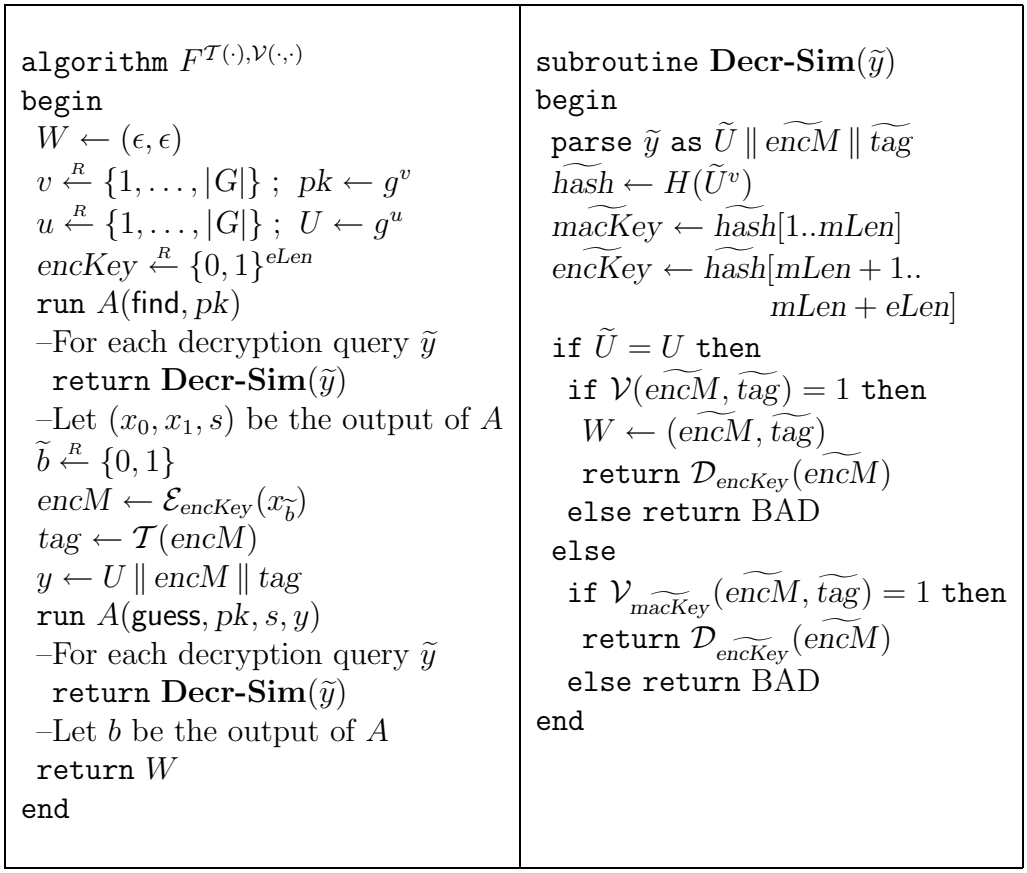

Figure II.7: *Algorithm F for attacking the security of* MAC.

hard-coreness of H for $\mathcal{G}$ under non-adaptive Diffie-Hellman attack, and F for the message authentication scheme. For this purpose, let y be the challenge ciphertext in experiment $\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$ and let SomeValid be the event where A makes a Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$ in this experiment. Let $\overline{\text{SomeValid}}$ denote the event where there is no Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$ in experiment $\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. We make use of the following three claims.

Claim II.G.2 $\Pr[\,\mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\mathsf{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2}$.

Proof: When the input $W = H(g^{uv})$, we notice that C is running A as the latter would be run in its attack on the adaptive chosen-ciphertext security of DHIES. Therefore, the claim follows from the definition of $\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. ∎

Claim II.G.3 $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\mathsf{H},C} = 1 \wedge \overline{\textsc{SomeValid}}\,] \leq \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0)}{2}$

Proof: When A does not make a Type 1 query to its decryption oracle nor makes a Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$, C runs A in the same way B does. Hence, the probability that C outputs 1 given $\overline{\textsc{SomeValid}}$ is at most $1/2 + 1/2 \cdot \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM},B}$. Since B makes 0 encryption queries and has time-complexity at most t, the claim follows directly from the assumed security of SYM. ∎

Claim II.G.4 $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\mathsf{H},C} = 1 \wedge \textsc{SomeValid}\,] \leq \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu)$

Proof: When there is a Type 1 query $\widetilde{y}$ to the decryption oracle such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$, let i be the number of one such query and let $y_i = U \,\|\, encM_i \,\|\, tag_i$ be its value. By assumption, we know that $\mathcal{V}_{macKey}(encM_i, tag_i) = 1$. Because $(encM_i, tag_i) \neq (encM, tag)$ (or otherwise $y_i = y$ and A would have queried its decryption oracle with the challenge ciphertext), either $encM_i$ was not queried of tag-generation oracle $\mathcal{T}(\cdot)$ or tag_i was not the response returned by tag-generation oracle $\mathcal{T}(\cdot)$ on query $encM$. In either case, $(encM_i, tag_i)$ is a valid forgery and F succeeds in breaking MAC. Therefore, $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\mathsf{H},C} = 1 \wedge \textsc{SomeValid}\,] \leq \Pr[\,\textsc{SomeValid}\,] \leq \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC},F}$. Hence, since F has time-complexity at most t, makes exactly one query to its tag-generation oracle $\mathcal{T}(\cdot)$ whose length is at most c (the upper bound on the length of challenge ciphertext), and makes at most q queries to its verification oracle $\mathcal{V}(\cdot,\cdot)$ the sum of whose lengths is at most μ bits, the claim follows from the assumed security of MAC. ∎

From Definition II.E.4 and Claims II.G.2, II.G.3, and II.G.4, we have that:

$$\begin{aligned}
&\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H},C} \\
&\quad\geq \quad \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2} - \frac{1}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0)}{2} - \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu) \\
&\quad= \quad \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0)}{2} - \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu)\ ;
\end{aligned}$$

whence

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A} \leq \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0) + 2 \cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H},C} + 2 \cdot \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu) \ .$$

We conclude that, since C has time-complexity at most t and makes at most q queries to its oracle $\mathcal{H}_v$, $\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H},C} \leq \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H}}(t,q)$. Thus, from the above, we have

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A} \ \leq \ \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0) + 2 \cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\mathsf{H}}(t,q) + 2 \cdot \mathbf{Adv}^{\text{mac}}_{\mathsf{MAC}}(t,1,c,q,\mu) \ .$$

But A was an arbitrary adversary subject to the constraint that it has time-complexity at most t and makes at most q queries to its decryption oracle the sum of whose lengths is at most μ bits, and the length of the challenge ciphertext is at most c in experiment $\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. The theorem follows. ∎

II.H ODH and SDH

In this section, we first exploit the relationship between the strong Diffie-Hellman (SDH) and the oracle Diffie-Hellman (ODH) assumptions when the hash function H is modeled as a random oracle. More specifically, we show that, in the random oracle (RO) model, the SDH assumption implies the ODH assumption. We then go on to prove a lower bound on the complexity of strong Diffie-Hellman assumption with respect to generic algorithms.

However, before proving the implication between the SDH and ODH assumption in the RO model, we need to back up a little and modify the experiment defining the security of ODH assumption to account for the presence of a random oracle H. The following is the definition of ODH assumption in the the RO model.

Definition II.H.1 [Oracle Diffie-Hellman in the random oracle model: ODH-RO] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let *hLen* be a number, let Ω be the set of all functions from $\{0,1\}^*$ to $\{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

$\texttt{experiment}\ \mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},\mathsf{H},A}$	$\texttt{experiment}\ \mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},\mathsf{H},A}$
$u \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ U \leftarrow g^u$	$u \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ U \leftarrow g^u$
$v \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ V \leftarrow g^v$	$v \xleftarrow{R} \{1,\ldots,\lvert G\rvert\}\ ;\ V \leftarrow g^v$
$W \leftarrow H(g^{uv})$	$W \xleftarrow{R} \{0,1\}^{hLen}$
$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$	$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$
$H \xleftarrow{R} \Omega$	$H \xleftarrow{R} \Omega$
$b \leftarrow A^{\mathcal{H}_v(\cdot),H(\cdot)}(U,V,W)$	$b \leftarrow A^{\mathcal{H}_v(\cdot),H(\cdot)}(U,V,W)$
$\texttt{return}\ b$	$\texttt{return}\ b$

Now define the *advantage* of A in violating the oracle Diffie-Hellman assumption in the random oracle model and the *advantage function* of ODH assumption in the RO model, respectively, as follows:

$$\begin{aligned} \mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\mathsf{H},A} &= \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},\mathsf{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},\mathsf{H},A} = 1\,] \\ \mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\mathsf{H}}(t,q_h,q_o) &= \max_A\{\,\mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\mathsf{H},A}\,\}\ , \end{aligned}$$

where the maximum is over all A with time-complexity t, making at most q_h queries to its H-oracle and q_o queries to its $\mathcal{H}_v$-oracle. Here A is not allowed to call its oracle $\mathcal{H}_v$ on input g^u. $\diamondsuit$

The following theorem shows that, in the random oracle (RO) model, the strong Diffie-Hellman assumption implies the oracle Diffie-Hellman assumption.

Theorem II.H.2 Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let the associated hash function H be chosen at random. Let q_h and q_o be, respectively, the total number of queries to H-oracle and to the $\mathcal{H}_v$-oracle. Then,

$$\mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\mathsf{H}}(t,q_h,q_o) \ \leq\ \mathbf{Adv}^{\text{sdh}}_{\mathcal{G}}(t + q_h\ q_o\ O(gLen + hLen), (q_h+q_o)^2)\ .$$

Proof: Let A be any adversary against the ODH assumption in the RO model. Let t be its time-complexity and q_h and q_o be, respectively, the number of queries it makes to H and $\mathcal{H}_v$ oracles. We can construct an adversary B for the Diffie-Hellman problem under SDH on $\mathcal{G}$, using A as a sub-routine, as follows.

ALGORITHM B. Algorithm B is shown in Figure II.8. B is given as input (U, V), where $U = g^u$ and $V = g^v$ for random u and v and outputs a value *guess*, its guess for g^{uv}. B is also given access to a restricted DDH oracle $\mathcal{O}_v$. Initially, B picks two values, $hash_0$ and $hash_1$, at random and set $hash_0$ to be the output of H on input U^v (although it still does not know the value of U^v). It also fix a default value g for *guess*. Then it runs A as a subroutine, feeding its input with either $hash_0$ or $hash_1$. Our hope is that, at some point, A is going to make a query of the form U^v to the H oracle, since otherwise A would have no advantage in distinguishing the real output $hash_0 \stackrel{\text{def}}{=} H(U^v)$ from the random string $hash_1$. To find out when A queries its H-oracle on input U^v, we query our restricted DDH oracle $\mathcal{O}_v$ on input (U, X) whenever A makes a query X to its H-oracle. If $\mathcal{O}_v(U, X)$ returns 1, then $X = U^v$ and we update the value of *guess* to X. Notice that B does not have access to either the H-oracle or the $\mathcal{H}_v$-oracle. However, it can simulate both oracles.

The oracle H is to be simulated as follows. In response to a query X, if X has been asked of H, then return the same response as given to that previous query. If X has not been asked of H, then first check whether $\mathcal{O}_v(U, X) = 1$. If so, then update the value of *guess* to X and return $hash_0$ as the response to the current H-query. If not, then check whether there was some query $\widetilde{U}$ to the $\mathcal{H}_v$-oracle such that $\mathcal{O}_v(\widetilde{U}, X) = 1$. If there was such query, then let *hash* be the same value used as response to that query. If not, then let *hash* be a random string. Return *hash* as the response to the current H query.

The $\mathcal{H}_v$-oracle is to be simulated as follows. When a query $\widetilde{U} \neq U$ is made, check whether $\widetilde{U}$ has already been asked of $\mathcal{H}_v$. If so, then return the same response as given to that previous query. If not, then check whether there was a previous H-query X where $\mathcal{O}_v(\widetilde{U}, X) = 1$. If so, then let *hash* be the output given to that H-query. If not, then let *hash* be a random value. Return *hash* as the response to the current query.

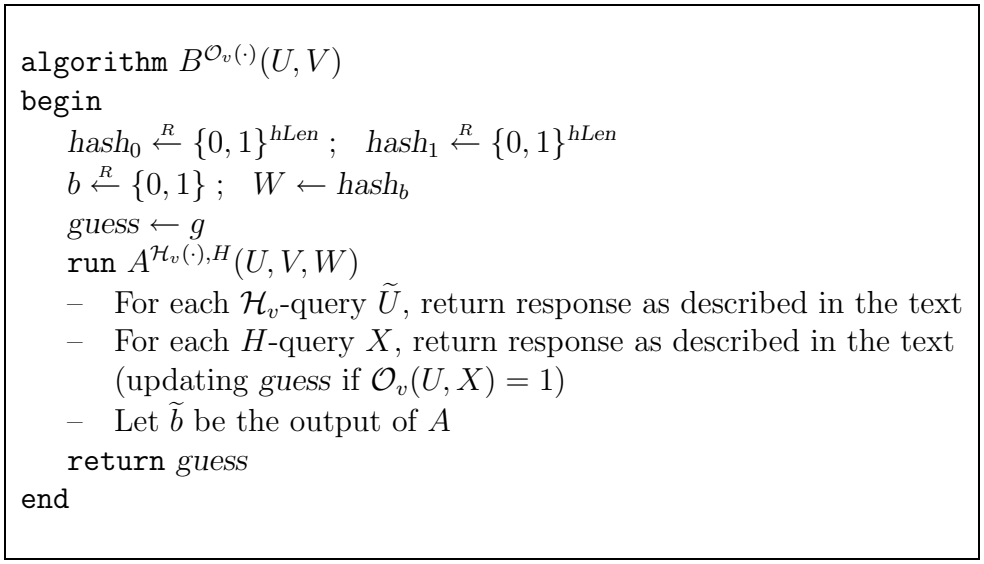

Figure II.8: *Algorithm B for attacking the hard-coreness of the Diffie-Hellman problem under SDH on $\mathcal{G}$.*

Notice that B runs in time at most $t + q_o q_h O(hLen + gLen)$ and makes at most $(q_o + q_h)^2$ queries to its $\mathcal{O}_v$-oracle.

Analysis. Consider the experiments $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A}$ and $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A}$ and let AskA and $\overline{\text{AskA}}$ denote, respectively, the event in which H-oracle query U^v is made by A and its complement.

When query U^v is not made directly by A to its H-oracle, there is no way for it to tell whether its input W is equal to $H(U^v)$ or a random string of length $hLen$ since the former can take any value in $\{0,1\}^{hLen}$. Hence, the probabilities that A outputs 1 in experiments $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A}$ and $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A}$, given that A does not query its H-oracle on input U^v are exactly the same. That is, $\Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1 \wedge \overline{\text{AskA}}\,]$ $= \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1 \wedge \overline{\text{AskA}}\,]$.

Consider now the case in which adversary A queries its H-oracle on input U^v in experiments $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A}$ and $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A}$. In such case, we know that adversary B, which runs A as a sub-routine, succeeds in solving the Diffie-Hellman problem under SDH on $\mathcal{G}$. This is because we know there is an index $i \in \{1, \ldots, q_h\}$ such

that $h_i = X$ and $X = U^v$. Since $\mathcal{O}_v(U, X) = 1$, *guess* will take the correct value U^v and B will succeed in solving the strong Diffie-Hellman problem on $\mathcal{G}$. Hence, $\Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] \leq \Pr[\,\text{AskA}\,] \leq \mathbf{Adv}^{\text{sdh}}_{\mathcal{G},B}$. Moreover, since B makes at most $(q_o + q_h)^2$ queries to its oracle $\mathcal{O}_v$ and has time complexity at most $t + q_o q_h O(hLen + gLen)$, it is also the case that $\mathbf{Adv}^{\text{sdh}}_{\mathcal{G},B} \leq \mathbf{Adv}^{\text{sdh}}_{\mathcal{G}}(t, (q_o + q_h)^2)$.

Putting it all together, we have that

$$
\begin{aligned}
&\mathbf{Adv}^{\text{odh}}_{\mathcal{G},H,A} \\
&\quad = \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1\,] \\
&\quad = \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] + \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1 \wedge \overline{\text{AskA}}\,] \\
&\qquad -\Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1 \wedge \overline{\text{AskA}}\,] \\
&\quad = \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A} = 1 \wedge \text{AskA}\,] \\
&\quad \leq \Pr[\,\text{AskA}\,] \\
&\quad \leq \mathbf{Adv}^{\text{sdh}}_{\mathcal{G},B} \\
&\quad \leq \mathbf{Adv}^{\text{sdh}}_{\mathcal{G}}(t + q_o q_h O(hLen + gLen), (q_o + q_h)^2).
\end{aligned}
$$

The bound claimed in the theorem follows easily from the fact that A was an arbitrary adversary subject to the constraint that it had time-complexity at most t and made at most q_h queries to its H-oracle and at most q_o queries to its $\mathcal{H}_v$-oracle. ∎

Lower bounds with respect to generic algorithms. Generic algorithms in groups are algorithms which do not make use of any special properties of the encoding of group elements other than assuming each element has a unique representation. This model was introduced by Shoup [75] and is very useful in proving lower bounds (with respect to such algorithms) for some problems. In fact, Shoup proved that in such a model both the discrete logarithm and the Diffie-Hellman problems are hard to solve as long as the order of the group contains at least one large prime factor. Following the same approach, we also use this model

here to prove lower bounds for some new problems we introduce. Let us proceed now with the formalization of this model.

Let $Z_n = \{1, \ldots, n\}$ be the additive group of integers modulo n, the order of the group. Let S be a set of bit strings of order at least n. We call an injective map from Z_n to S an *encoding function*. One example for such a function would be the function taking $u \in Z_{|G|}$ to $g^{u \bmod |G|}$, where G is a finite cyclic group of order $|G|$ generated by the group element g.

A generic algorithm is a probabilistic algorithm A which takes as input a list

$$(\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_k)),$$

where each $x_i \in Z_n$ and σ is a random *encoding function*, and outputs a bit string. During its execution, A can make queries to an oracle Σ. Each query will result in updating the encoding list, to which A has always access. Σ gets as input two indices i and j and sign bit, and then computes $\sigma(x_i \pm x_j)$ and appends it to the list. It is worth noticing that A does not depend on σ, since it is only accessible by means of oracle queries.

We need to extend the original generic model to allow queries to the restricted DDH oracle $\mathcal{O}_v$. In this case, $\mathcal{O}_v$ gets as input two indices i and j and returns 1 if $x_j = v \cdot x_i$ and 0, otherwise. In general lines, our result shows that the restricted DDH oracle $\mathcal{O}_v$ does not help in solving the Diffie-Hellman problem whenever the group order contains a large prime factor. One should note, however, that our result has no implications on non-generic algorithms, such as index-calculus methods for multiplicative groups of integers modulo a large prime. Let us state this more formally.

Definition II.H.3 [SDH in generic model] Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n, and let σ be a random encoding function of Z_n on S. In addition, let Ω be the set of all mappings Z_n to S. Let A be an generic algorithm. Consider the experiment

$\texttt{experiment}\ \mathbf{Exp}_n^{\text{g-sdh}}$

$\sigma \xleftarrow{R} \Omega$

$g \leftarrow \sigma(1)$

$u \xleftarrow{R} \{1, \ldots, n\}\ ;\ U \leftarrow \sigma(u)$

$v \xleftarrow{R} \{1, \ldots, n\}\ ;\ V \leftarrow \sigma(v)$

$\Sigma(i, j, \pm) \stackrel{\text{def}}{=} x_i \pm x_j$

$\mathcal{O}_v(i, j) \stackrel{\text{def}}{=} (x_j = v x_i)$

$Z \leftarrow A^{\mathcal{O}_v(\cdot,\cdot),\Sigma(\cdot,\cdot,\cdot)}(g, U, V)$

$\texttt{if}\ Z = \sigma(uv)\ \texttt{then}\ b \leftarrow 1\ \texttt{else}\ b \leftarrow 0$

$\texttt{return}\ b$

Now define the *advantage* of A in violating the strong Diffie-Hellman assumption in the generic model and the advantage function of SDH assumption in this model, respectively, as follows:

$$\begin{aligned} \mathbf{Adv}_{n,A}^{\text{g-sdh}} &= \Pr[\,\mathbf{Exp}_{n,A}^{\text{g-sdh}} = 1\,] \\ \mathbf{Adv}_{n}^{\text{g-sdh}}(q) &= \max_{A}\{\,\mathbf{Adv}_{n,A}^{\text{g-sdh}}\,\}\ , \end{aligned}$$

where q is the total number of queries made by A to its oracles. $\diamondsuit$

Theorem II.H.4 Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n. Then, for any number q,

$$\mathbf{Adv}_{n}^{\text{g-sdh}}(q) \leq O(q^2/p)$$

where p is the largest prime factor of n.

A corollary of Theorem II.H.4 is that any generic algorithm solving the Diffie-Hellman problem under SDH with success probability bounded away from 0 has to perform at least $\Omega(p^{1/2})$ group operations.

Proof: Here we just present a proof sketch using a technique used by Shoup in [75]. Let $n = s\ p^t$ with $\gcd(s, p) = 1$. Since additional information only reduces the running time, we can assume that solving the Diffie-Hellman problem in the subgroup of order s is easy. Hence, let $n = p^t$ wlog.

We start by running algorithm A. In doing so, we need to simulate all its oracles. We play the following game. Let U and V be indeterminants. During the execution of the algorithm, we will maintain a list $F_1, \ldots, F_k$ of polynomials in $Z_{p^t}[U, V]$, along with a list $\sigma_1, \ldots, \sigma_k$ of distinct values in S. Initially, we have $F_1 = 1$, $F_2 = U$, and $F_3 = V$; and three distinct values σ_1, σ_2, and σ_3 chosen at random from S. When the algorithm makes a query $(i, j, \pm)$ to its Σ-oracle, we first compute $F_{k+1} = F_i \pm F_j \in Z_{p^t}[U, V]$ and check whether there is some $l \leq k$ such that $F_{k+1} = F_l$. If so, then we return σ_l to A. Else we pick choose a random but distinct σ_{k+1}, return it to A, and update both lists. When the algorithm makes a query (i, j) to its $\mathcal{O}_v$, we return 1 if $F_j = V \cdot F_i$ else 0.

We can assume that A outputs an element in the encoding list (otherwise $\mathbf{Adv}^{\mathrm{sdh}}_{n,A} \leq 1/(p-q)$), where q is the number of queries made by A. Then, let us choose u and v at random from Z_{p^t}. Notice that $\mathbf{Adv}^{\mathrm{sdh}}_{n,A}$ can be upper bounded by the probability of one of the following happening: $F_i(u, v) = F_j(u, v)$ for some F_i and F_j; or $F_i(u, v) = uv$ for some i; or $F_j \neq V_i$ but $F_j(u, v) = vF_i(u, v)$. Otherwise, the algorithm cannot learn anything about u or v except that $F_i(u, v) \neq F_j(u, v)$ for every i and j. But, using results from [75], for fixed i and j, the probability that $F_i - F_j$ vanishes is at most $1/p$; the probability that $F_i - UV$ vanishes is at most $2/p$; and the probability that $F_j - VF_i$ vanishes is at most $2/p$. It follows that the probability of one of these happening is $O(q^2/p)$. The theorem follows from the fact that A was an arbitrary adversary subject to the constraint that it makes at most q queries to its oracles. ∎

Chapter III

Re-keyed Encryption Schemes

III.A Introduction

Re-keying (also called key-derivation) is a commonly employed paradigm in computer security systems, about whose security benefits users appear to have various expectations. Yet the security of these methods has not been systematically investigated. Let us begin with some examples that illustrate usage, commonly employed implementations, and motivation for re-keying, and see what security issues are raised. We then go on to our results.

Re-keyed encryption. Say two parties share a key K, and want to encrypt data they send to each other. They will use some block cipher based mode of operation, say CBC. The straightforward approach is to use K directly to encrypt the data. An often employed alternative is re-keyed encryption. The key K is not used to encrypt data but rather viewed as a master key. Subkeys $K_1, K_2, K_3, \ldots$ are derived from K, by some process called the re-keying process. A certain number l of messages are encrypted using K_1 and then the parties switch to K_2. Once l messages have been encrypted under K_2 they switch to K_3 and so on.

Examples of re-keying methods. Many different re-keying methods are possible. Let us outline the two most commonly used. In each case $F(\cdot, \cdot)$ is

a map that takes a k-bit key κ and k-bit input x to a k-bit output $F(\kappa, x)$. (This might be implemented via a block cipher or a keyed hash function.) The *parallel* method consists of setting $K_i = F(K, i)$ for $i = 1, 2, \ldots$. The *serial* method sets $k_0 = K$ and then sets $K_i = F(k_{i-1}, 0)$ and $k_i = F(k_{i-1}, 1)$ for $i = 1, 2, \ldots$. For a pictorial representation of these methods, please refer to Figure III.1. Many other methods are possible, including hybrids of these two such as tree-based re-keying (see Section III.C).

WHY RE-KEY? Common attacks base their success on the ability to get lots of encryptions under a single key. For example differential or linear cryptanalysis [19, 59] will recover a DES key once a certain threshold number of encryptions have been performed using it. Furthermore, most modes of operation are subject to birthday attacks [8], leading to compromise of the privacy of a scheme based on a block cipher with block size k once $2^{k/2}$ encryptions are performed under the same key. Typically, the birthday threshold is lower than that of the cryptanalytic attacks.

Thus, if encryption is performed under a single key, there is a certain maximum threshold number of messages that can be safely encrypted. Re-keying protects against attacks such as the above by changing the key before the threshold number of encryptions permitting the attack is reached. It thus effectively extends the lifetime of the (master) key, increasing the threshold number of encryptions that can be performed without requiring a new exchange of keys.

QUESTIONS. Although re-keying is common practice, its security has not been systematically investigated. We are interested in the following kinds of questions. Does re-keying really work, in the sense that there is some *provable* increase in security of an application like re-keyed encryption described above? That is, can one prove that the encryption threshold —number of messages of some fixed length that can be safely encrypted— increases with re-keying? How do different re-keying processes compare in terms of security benefits? Do some offer more security than others? How frequently should the key be changed, meaning

how should one choose the parameter l given the parameters of a cryptographic system?

HIGH LEVEL ANSWERS. At the highest level, our answer to the most basic question (does re-keying increase security?) is "YES." We are able to justify the prevailing intuition with concrete security analyses in the provable security framework and show that re-keying, properly done, brings significant security gains in practical situations, including an increase in the encryption threshold. Seen from closer up, our results give more precise and usable information. We quantify the security provided by different re-keying processes as a function of the security of the primitives they use. This enables comparison between these processes. Thus, say a user wants to encrypt a certain amount of data with a block cipher of a certain strength: our results can enable this user to figure out which re-keying scheme to use, with what parameters, and what security expectations.

RE-KEYED CBC ENCRYPTION. As a sample of our results we discuss CBC encryption. Suppose we CBC encrypt with a block cipher F having key-length and block-length k. Let's define the encryption threshold as the number Q of k-bit messages that can be safely encrypted. We know from [8] that this value is $Q \approx 2^{k/2}$ for the single-key scheme. We now consider re-keyed CBC encryption under the parallel or serial re-keying methods discussed above where we use the same block cipher F as the re-keying function. We show that by re-keying every $2^{k/3}$ encryptions —i.e. set the subkey lifetime $l = 2^{k/3}$— the encryption threshold increases to $Q \approx 2^{2k/3}$. That is, one can safely encrypt significantly more data by using re-keying. The analysis can be found in Section III.D.

OVERVIEW OF APPROACH AND RESULTS. Re-keying can be used in conjunction with any shared-key based cryptographic data processing. This might be data encryption, under any of the common modes of operation; it might be data authentication using some MAC; it might be something else. We wish to provide tools that enable the analysis of any of these situations. So rather than analyze

each re-keyed application independently, we take a modular approach. We isolate the re-keying process, which is responsible for producing subkeys based on a master key, from the application which uses the subkeys. We then seek a general security attribute of the re-keying process which, if present, would enable one to analyze the security of any re-keying based application. We suggest that this attribute is pseudorandomness. We view the re-keying process as a stateful pseudorandom bit generator and adopt a standard notion of security for pseudorandom bit generators [21, 81]. We measure pseudorandomness quantitatively, associating to any re-keying process (stateful generator) $\mathcal{G}$ an advantage function $\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G},n}(t)$, which is the maximum probability of being able to distinguish n output blocks of the generator from a random string of the same length when the distinguishing adversary has running time at most t. We then analyze the parallel and serial generators, upper bounding their advantage functions in terms of an advantage function associated to the underlying primitive F. See Section III.B.

To illustrate an application, we then consider re-keyed symmetric encryption. We associate a re-keyed encryption scheme to any base symmetric encryption scheme (e.g. CBC) and any generator. We show how the advantage function of the re-keyed encryption scheme can be bounded in terms of the advantage function of the base scheme and the advantage function of the generator. (The advantage function of an encryption scheme, whether the base or re-keyed one, measures the breaking probability as a function of adversary resources under the notion of left-or-right security of [8].) Coupling our results about the parallel and serial generators with known analyses of CBC encryption [8] enables us to derive conclusions about the encryption threshold for CBC as discussed above. See Section III.D.

Security of the parallel and serial generators. Our analysis of the parallel and serial generators as given by Theorems III.B.4 and III.B.5 indicates that their advantage functions depend differently on the advantage function of the underlying primitive F. (We model the latter as a pseudorandom function [43] and associate an advantage function as per [11].) In general, the parallel gen-

erator provides better security. This is true already when F is a block cipher but even more strikingly the case when F is a non-invertible PRF. This should be kept in mind when choosing between the generators for re-keying. However, whether or not it eventually helps depends also on the application. For example, with CBC encryption, there is no particular difference in the quantitative security providing by parallel and serial re-keying (even though both provide gains over the single-key scheme). This is due to the shape of the curve of the advantage function of the base CBC encryption function as explained in Section III.D.

Forward security. Another possible motivation for re-keying is to provide forward security. The goal here is to minimize the amount of damage that might be caused by key exposure due, for instance, to compromise of the security of the underlying system storing the secret key. (Forward security was first considered for session keys [46, 33] and then for digital signatures [13].) Under re-keying, the adversary would only get the current subkey and state of the system. It could certainly figure out all future subkeys, but what about past ones? If the re-keying process is appropriately designed, it can have forward security: the past subkeys will remain computationally infeasible for the adversary to derive even given the current subkey and state, and thus ciphertexts that were formed under them will not be compromised. It is easy to see that the parallel generator does not provide forward security. It can be shown however that the serial one does. A treatment of forward security in the symmetric setting, including a proof of the forward security of the serial generator and the corresponding re-keyed encryption scheme, can be found in [18].

Related work. Another approach to increasing the encryption threshold, discussed in [12], is to use a mode of encryption not subject to birthday attack (e.g. CTR rather than CBC) and implement this using a non-invertible, high security PRF rather than a block cipher. Constructions of appropriate PRFs have been provided in [12, 48]. Re-keying is cheaper in that one can use the given block cipher and a standard mode like CBC and still push the encryption threshold well

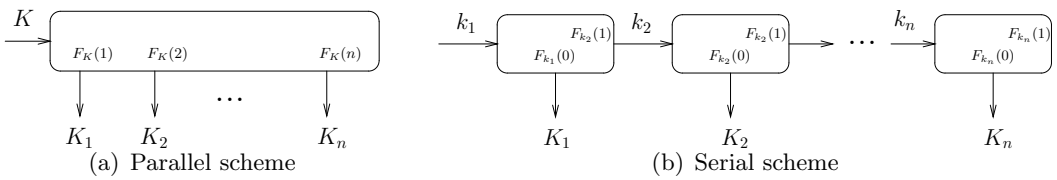

Figure III.1: The parallel and serial re-keying schemes.

beyond the birthday threshold.

Re-keying requires that parties maintain state. Stateless methods of increasing security beyond the birthday bound are discussed in [10].

These results are based on a previous joint work with Mihir Bellare [1].

III.B Re-keying processes as pseudorandom generators

The subkeys derived by a re-keying process may be used in many different ways: data encryption or authentication are some but not all of these. To enable modular analysis, we separate the subkey generation from the application that uses the subkeys. We view the re-keying process —which generates the subkeys— as a stateful pseudorandom bit generator. In this section we provide quantitative assessments of the security of various re-keying schemes with regard to notions of security for pseudorandom generators. These application independent results are used in later sections to assess the security of a variety of different applications under re-keying.

Stateful generators. A stateful generator $\mathcal{G} = (\mathcal{K}, \mathcal{N})$ is a pair of algorithms. The probabilistic *key generation* algorithm $\mathcal{K}$ produces the initial state, or seed, of the generator. The deterministic *next step* algorithm $\mathcal{N}$ takes the current state as input and returns a block, viewed as the output of this stage, and an updated state, to be stored and used in the next invocation. A sequence $Out_1, Out_2, \ldots$ of pseudorandom blocks is defined by first picking an initial seed $St_0 \leftarrow \mathcal{K}$ and then iterating: $(Out_i, St_i) \leftarrow \mathcal{N}(St_{i-1})$ for $i \geq 1$. (When the generator is used for re-keying, these are the subkeys. Thus Out_i was denoted K_i

in Section III.A). We assume all output blocks are of the same length and call this the block length.

We now specify two particular generators, the parallel and serial ones. We fix a PRF F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$. (As the notation indicates, we are making the simplifying assumption that the key length, as well as the input and output lengths of each individual function $F(K,\cdot)$ are all equal to k.) In practice, this might be instantiated via a block cipher or via a keyed hash function such as HMAC [7]. (For example, if DES is used, then we set $k = 64$ and define $F(K,\cdot)$ to be $\mathrm{DES}(K[1..56],\cdot)$.)

Construction III.B.1 (Parallel generator) The F-based parallel generator $\mathcal{PG}[F] = (\mathcal{K},\mathcal{N})$ is defined by

$\texttt{algorithm}\ \mathcal{K}$	$\texttt{algorithm}\ \mathcal{N}(\langle i, K\rangle)$
$K \xleftarrow{R} \{0,1\}^k$	$Out \leftarrow F(K,i)$
Return $\langle 0, K\rangle$	Return $(Out, \langle i+1, K\rangle)$

The state has the form $\langle i, K\rangle$ where K is the initial seed and i is a counter, initially zero. In the i-th stage, the output block is obtained by applying the K-keyed PRF to the (k-bit binary representation of the integer) i, and the counter is updated. This generator has block length k. ▮

Construction III.B.2 (Serial generator) The F-based serial generator $\mathcal{SG}[F]$ $= (\mathcal{K},\mathcal{N})$ is defined by

$\texttt{algorithm}\ \mathcal{K}$	$\texttt{algorithm}\ \mathcal{N}(K)$
$K \xleftarrow{R} \{0,1\}^k$	$Out \leftarrow F(K,0)$
Return K	$K \leftarrow F(K,1)$
	Return (Out, K)

The state is a key K. In the i-th stage, the output block is obtained by applying the K-keyed PRF to the (k-bit binary representation of the integer) 0, and the new state is a key generated by applying the K-keyed PRF to the (k-bit binary representation of the integer) 1. This generator has block length k. ▮

PSEUDORANDOMNESS. The standard desired attribute of a (stateful) generator is pseudorandomness of the output sequence. We adopt the notion of [21, 81] which formalizes this by asking that the output of the generator on a random seed be computationally indistinguishable from a random string of the same length. Below, we concretize this notion by associating to any generator an advantage function which measures the probability that an adversary can detect a deviation in pseudorandomness as a function of the amount of time invested by the adversary.

Definition III.B.3 (Pseudorandomness of a stateful generator) Let $\mathcal{G} = (\mathcal{K}, \mathcal{N})$ be a stateful generator with block length k, let n be an integer, and let A be an adversary. Consider the experiments

$\texttt{experiment}\ \mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,A}$	$\texttt{experiment}\ \mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,A}$
$St_0 \leftarrow \mathcal{K}\ ;\ s \leftarrow \varepsilon$	$s \leftarrow \{0,1\}^{n \cdot k}$
$\texttt{for}\ \ i = 1, \ldots, n\ \texttt{do}$	$g \leftarrow A(s)$
$\quad (Out_i, St_i) \leftarrow \mathcal{N}(St_{i-1})\ ;\ s \leftarrow s \,\|\, Out_i$	$\texttt{return}\ g$
$g \leftarrow A(s)$	
$\texttt{return}\ g$	

Now define the *advantage* of A and the *advantage function of the generator*, respectively, as follows:

$$\begin{aligned} \mathbf{Adv}^{\text{prg}}_{\mathcal{G},n,A} &= \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,A} = 1\,] \\ \mathbf{Adv}^{\text{prg}}_{\mathcal{G},n}(t) &= \max_{A} \{\, \mathbf{Adv}^{\text{prg}}_{\mathcal{G},n,A} \,\}\ , \end{aligned}$$

where the maximum is over all A with "time-complexity" t. ▮

Here "time-complexity" is the maximum of the execution times of the two experiments plus the size of the code for A, all in some fixed RAM model of computation. (Note that the execution time refers to that of the entire experiment, not just the execution time of the adversary.) The advantage function is the maximum likelihood of the security of the pseudorandom generator $\mathcal{G}$ being compromised by an adversary using the indicated resources.

Security measure for PRFs. Since the security of the above constructions depends on that of the underlying PRF F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$, we recall the measure of [11], based on the notion of [43]. Let R^k denote the family of all functions mapping $\{0,1\}^k$ to $\{0,1\}^k$, under the uniform distribution. If D is a distinguisher having an oracle, then

$$\mathbf{Adv}^{\mathrm{prf}}_{F,D} \;=\; \Pr[\, D^{F(K,\cdot)} = 1 \;:\; K \xleftarrow{R} \{0,1\}^k \,] - \Pr[\, D^{f(\cdot)} = 1 \;:\; f \xleftarrow{R} R^k \,]$$

is the advantage of D. The advantage function of F is

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(t,q) \;=\; \max_{D} \{\, \mathbf{Adv}^{\mathrm{prf}}_{F,D} \,\} \;,$$

where the maximum is over all A with "time-complexity" t and making at most q oracle queries. The time-complexity is the execution time of the experiment $K \xleftarrow{R} \{0,1\}^k \;;\; v \leftarrow D^{F(K,\cdot)}$ plus the size of the code of D, and, in particular, includes the time to compute $F_K(\cdot)$ and reply to oracle queries of D.

Pseudorandomness of the parallel and serial generators. The following two theorems show how the pseudorandomness of the two generators is related to the security of the underlying PRF.

Theorem III.B.4 Let F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF and let $\mathcal{PG}[F]$ be the F-based parallel generator defined in Construction III.B.1. Then

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n}(t) \le \mathbf{Adv}^{\mathrm{prf}}_{F}(t,n) \;. \blacksquare$$

Proof: Let A be an adversary attacking the pseudorandomness of $\mathcal{PG}[F]$ and t be the maximum of the running times of $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n,A}$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n,A}$. We want to upper bound $\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n,A}$. We do so by constructing a distinguisher D for F and relating its advantage to that of A. D has access to an oracle $\mathcal{O}$. It simply computes $s = \mathcal{O}(1) \,\|\, \ldots \,\|\, \mathcal{O}(n)$ and outputs the same guess as A on input s. We can see that when the oracle $\mathcal{O}$ is drawn at random from the family F, the probability that D returns 1 equals the probability that the experiment $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n,A}$ returns 1. Likewise, the probability that the experiment $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n,A}$ returns 1 equals that

of D returning 1 when $\mathcal{O}$ is drawn at random from the family of random functions R^k. As D runs in time at most t and makes exactly n queries to its oracle, we get that

$$\mathbf{Adv}^{\text{prg}}_{\mathcal{PG}[F],n,A} \;\leq\; \mathbf{Adv}^{\text{prf}}_{F}(t,n)\ .$$

Since A was an arbitrary adversary and the maximum of the running times of experiments $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{PG}[F],n,A}$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{PG}[F],n,A}$ is t, we obtain the conclusion of the theorem. ∎

Theorem III.B.5 Let $F\colon\ \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF and let $\mathcal{SG}[F]$ be the F-based parallel generator defined in Construction III.B.2. Then

$$\mathbf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n}(t) \leq n \cdot \mathbf{Adv}^{\text{prf}}_{F}(t+\log n, 2)\ .\blacksquare$$

Proof: Let A be an adversary attacking the pseudorandomness of $\mathcal{SG}[F]$ and t be the maximum of the running times of $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n,A}$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n,A}$. We want to upper bound $\mathbf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n,A}$. We begin by defining the following sequence of hybrid experiments, where j varies between 0 and n.

```
experiment Hybrid_{A,j}
    St ←R {0,1}^k ; s ← ε
    for i = 1, ..., n do
        if i ≤ j then Out_i ←R {0,1}^k
        else (Out_i, St) ← N(St)
        s ← s || Out_i
    g ← A(s)
    return g
```

Let P_j be the probability that experiment $\mathbf{Hybrid}_{A,j}$ returns 1, for $j = 0,\ldots,n$. Note that the experiments $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n,A}$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n,A}$ are identical to $\mathbf{Hybrid}_{A,0}$ and $\mathbf{Hybrid}_{A,n}$, respectively. (Not syntactically, but semantically.)

This means that $P_0 = \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n,A} = 1\,]$ and $P_n = \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n,A} = 1\,]$. Putting it all together, we have

$$\begin{aligned}\mathbf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n,A} &= \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{SG}[F],n,A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{SG}[F],n,A} = 1\,] \\ &= P_0 - P_n\ . \end{aligned} \tag{III.1}$$

We now claim that

$$\mathbf{Adv}^{\text{prg}}_{\mathcal{SG}[F],n,A} = P_0 - P_n \ \ \leq \ \ n \cdot \mathbf{Adv}^{\text{prf}}_{F}(t + \log n, 2)\ . \tag{III.2}$$

Since A was an arbitrary adversary, we obtain the conclusion of the theorem. It remains to justify Equation (III.2). We will do this using the advantage function of F. Consider the following distinguisher for F.

```
algorithm D^O
    j <-R {1,...,n} ; s <- ε
    for i = 1,...,n do
        if i < j then Out_i <-R {0,1}^k
        if i = j then Out_i <- O(0) ; St <- O(1)
        if i > j then (Out_i, St) <- N(St)
        s <- s || Out_i
    g <- A(s)
    return g
```

Suppose the oracle given to D was drawn at random from the family F. Then, the probability that D returns 1 equals the probability that expirement $\mathbf{Hybrid}_{A,j-1}$ returns 1, where j is the value chosen at random by D in its first step. Similarly, if the given oracle is drawn at random from the family of random functions R^k, then the probability that D returns 1 equals the probability that the experiment $\mathbf{Hybrid}_{A,j}$ returns 1, where j is the value chosen at random by D in its first step.

Hence,

$$\begin{aligned} \Pr[\, D^{\mathcal{O}} \,|\, \mathcal{O} \xleftarrow{R} F \,] &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \\ \Pr[\, D^{\mathcal{O}} \,|\, \mathcal{O} \xleftarrow{R} R^k \,] &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j} \ . \end{aligned}$$

Subtract the second sum from the first and exploit the collapse to get

$$\frac{P_0 - P_n}{n} \;=\; \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \;-\; \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j} \;=\; \mathbf{Adv}^{\mathrm{prf}}_{F,D} \ .$$

Note that D runs in time at most $t + O(\log n)$ and makes exactly 2 queries to its oracle, whence we get Equation (III.2). This concludes the proof of the theorem. ∎

The qualitative interpretation of Theorems III.B.4 and III.B.5 is the same: both the parallel and the serial generator are secure pseudorandom bit generators if the PRF is secure. The quantitative statements show however that the pseudorandomness of n output blocks depends differently on the security of the PRF in the two cases. For the parallel generator, it depends on the security of the PRF under n queries. For the serial generator, it depends on the security of the PRF against only a constant number of queries, but this term is multiplied by the number of output blocks. Comparing the functions on the right hand side in the two theorems will tell us which generator is more secure.

Examples. As an example, assume F is a block cipher. Since F is a cipher, each map $F(K, \cdot)$ is a permutation, and birthday attacks can be used to distinguish F from the family of random functions with a success rate growing as $q^2/2^k$ for q queries (c.f.. [11, Proposition 2.4]). Let us make the (heuristic) assumption that this is roughly the best possible, meaning

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(t, q) \approx \frac{q^2 + t}{2^k} \tag{III.3}$$

for t small enough to prevent cryptanalytic attacks. Now the above tells us that the advantage functions of the two generators grow as follows:

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n}(t) \approx \frac{n^2 + t}{2^k} \text{ and } \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(t) \approx \frac{nt}{2^k} \ .$$

Since $t \geq n$, the two functions are roughly comparable, but in fact the first one

has a somewhat slower growth because we would expect that $t \gg n$. So, in this case, the parallel generator is somewhat better.

Now assume F is not a block cipher but something that better approximates a random function, having security beyond the birthday bound. Ideally, we would like something like

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(t,q) \approx \frac{q+t}{2^k} \tag{III.4}$$

for t small enough to prevent cryptanalytic attacks. This might be achieved by using a keyed hash function based construction, or by using PRFs constructed from block ciphers as per [12, 48]. In this case we would get

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PG}[F],n}(t) \approx \frac{n+t}{2^k} \text{ and } \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{SG}[F],n}(t) \approx \frac{nt}{2^k}\ .$$

Thinking of $t \approx n$ (it cannot be less but could be more, so this is an optimistic choice), we see that the first function has linear growth and the second has quadratic growth, meaning the parallel generator again offers better security, but this time in a more decisive way.

These examples illustrate how the quantitative results of the theorems can be coupled with cryptanalytic knowledge or assumptions about the starting primitive F to yield information enabling a user to choose between the generators.

III.C Generalization: Tree-based re-keying

In this section, we suggest a more general way to generate keys based on a balanced tree. The idea is to start from secure but limited stateful pseudorandom number generator and build a more general and flexible stateful pseudorandom number generator, while still preserving security. In our tree-based construction of a stateful generator, each internal node represents a single stateful pseudorandom number generator and each leaf represents an output block (i.e., the key for the re-keyed symmetric encryption scheme) of the overall scheme at a certain stage. Except for those generators at the lowest level, each output block of a generator will feed the input (its coins) of the key generation algorithm of those generators

one level below which are directly connected to them in the tree (its children). The state $St = \langle St_1, \ldots, St_{L-1}, i \rangle$ will consist of the state of all generators in the path from the current leaf to the root $(St_1, \ldots, St_{L-1})$, where L is the total number of levels in the tree, plus the number of the current leaf, i. To obtain a new output block (the next leaf), we only need to obtain the output blocks of those generators which are not in the common path from both the current and next leaves to the root.

In order to be more flexible, we allow different values of arity at different levels, but we assume their values to be the same within each level. Wlog, we also assume that the length of the initial seed (actually the coins) of all generators at the same level to be the same and equal to length of the output block of their parents. We consider the root level to be 1.

III.C.1 Construction

Let us now specify our construction more formally. Refer to Figure III.2 for a pictorial representation. Let $\mathcal{G}_l = (\mathcal{K}_l, \mathcal{N}_l)$ be a stateful pseudorandom number generator used at level l. Let a_l be the arity of the tree at level l and k_l be the key length of $\mathcal{G}_l$. Then we define our overall stateful pseudorandom number generator $\mathcal{G} = (\mathcal{K}, \mathcal{N})$, whose key length is $k = k_1$, as follows:

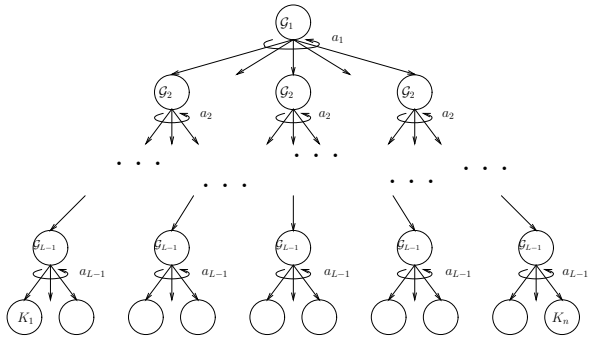

Figure III.2: Diagram of our tree-based construction

```
algorithm K
    l ← 1
    St_l ← K_l
    while l < L − 1 do
        (Out_l, St_l) ← N_l(St_l)
        St_{l+1} ← K_{l+1}(Out_l)
        l ← l + 1
    St ← ⟨St_1, ..., St_{L−1}, 0⟩
    return St
```

```
algorithm N(St)
    parse St as ⟨St_1, ..., St_{L−1}, i⟩
    l ← L − 1 ; d ← i + 1
    while d mod a_l = 0 do
        d ← ⌊d/a_l⌋ ; l ← l − 1
    (Out_l, St_l) ← N_l(St_l)
    while l < L − 1 do
        l ← l + 1 ; St_l ← K_l(Out_{l−1})
        (Out_l, St_l) ← N_l(St_l)
    St ← ⟨St_1, ..., St_{L−1}, i + 1⟩
    return (Out_{L−1}, St)
```

III.C.2 Security

We claim that the stateful pseudorandom number generator $\mathcal{G}$ we constructed in the previous subsection is secure as long as each underlying stateful pseudorandom number generator $\mathcal{G}_1, \ldots, \mathcal{G}_{L-1}$ is secure.

Theorem III.C.1 Let each $\mathcal{G}_l = (\mathcal{K}_l, \mathcal{N}_l)$ be a secure stateful generator with block size k_l for $l = 1, \ldots, L-1$. Define $a_0 = 1$ and let $n_l = \prod_{j=0}^{l-1} a_l$ be the total number of nodes at level l. Let $\mathcal{G}$ be the overall stateful generator formed out of the basic

stateful generators as above. Then

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G}}(n,t) \;\leq\; \sum_{l=1}^{L-1} n_l \;\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G}_l}(a_l,t)\;.$$

Proof:

The analysis here follows that of Goldreich, Goldwasser and Micali's construction of a pseudorandom function from a pseudorandom generator [43], and its later extensions such as the analysis of the cascade construction of pseudorandom functions [7]. We note however that although the analysis ideas are similar, our setting and conclusions are quite different. We are building and analyzing key derivation processes (stateful generators), not pseudorandom functions. More importantly, we prove that re-keying can actually increase the security of an application.

Let A be an adversary attacking the security of $\mathcal{G}$ and having a running time of at most t. We want to upper bound $\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G},A}$. Let $n_l = \prod_{i=0}^{l-1} a_i$ be the total number of nodes at level l ($a_0 = 1$). We begin by defining the sequence of hybrid experiments, $\mathbf{Hybrid}_{A,l,j}$, where l varies between 1 and $L-1$ and j varies between 0 and n_l. In experiment $\mathbf{Hybrid}_{A,l,j}$, the output of all generators in levels 1 up to $l-1$ and those in level l whose indices are smaller than j are replaced by truly random strings of the same length.

`experiment` $\mathbf{Hybrid}_{A,l,j}$ $\quad (1 \leq l \leq L-1) \quad (0 \leq j \leq n_l)$

$c_l \leftarrow 0$

`if` $c_l < j$ `then`

$Out_h \xleftarrow{R} \{0,1\}^{k_{h+1}}$

`else`

$Out_{h-1} \xleftarrow{R} \{0,1\}^{k_h}\ ;\ St_h \leftarrow \mathcal{K}_h(Out_{h-1})$

$(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$

`for` $h = l+1, \ldots, L-1$ `do`

$St_h \leftarrow \mathcal{K}_h(Out_{h-1})$

$(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$

$s \leftarrow Out_{L-1}$; $c_L \leftarrow 0$
`for` $c_L = 1, \ldots, L-1$ `do`
  $d \leftarrow c_L$; $h \leftarrow L-1$
  `while` $d \bmod a_h = 0$ `and` $h \geq l$ `do`
    $d \leftarrow \lfloor d/a_h \rfloor$; $h \leftarrow h-1$
  `if` $h = l-1$ `then`
    $c_l \leftarrow c_l + 1$; $h \leftarrow h+1$
    `if` $c_l < j$ `then`
      $Out_h \xleftarrow{R} \{0,1\}^{k_{h+1}}$
    `else`
      $Out_{h-1} \xleftarrow{R} \{0,1\}^{k_h}$; $St_h \leftarrow \mathcal{K}_h(Out_{h-1})$
      $(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$
  `else if` $h = l$ `and` $c_l < j$ `then`
    $Out_h \xleftarrow{R} \{0,1\}^{k_{h+1}}$
  `else`
    $(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$
  `while` $h < L-1$ `do`
    $h \leftarrow h+1$; $St_h \leftarrow \mathcal{K}_h(Out_{h-1})$
    $(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$
  $s \leftarrow s \,\|\, Out_{L-1}$
$g \leftarrow A(\mathsf{guess}, s)$
`return` g

Let $P_{l,j}$ be the probability that experiment $\mathbf{Hybrid}_{A,l,j}$ returns 1, for $l = 1, \ldots, L$ and $j = 0, \ldots, n_l$. We first notice that experiments $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,A}$ and $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,A}$ are respectively identical, semantically, to experiments $\mathbf{Hybrid}_{A,1,0}$ and $\mathbf{Hybrid}_{A,L,n_L}$. As a result,

$$\begin{aligned} \mathbf{Adv}^{\text{prg}}_{\mathcal{G},A} &= \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,A} = 1\,] \\ &= P_{1,0} - P_{L,n_L}\ . \end{aligned} \tag{III.5}$$

Moreover, we can still go a bit further and notice that for $l = 1, \ldots, L-2$, $P_{l,n_l} = P_{l+1,0}$. Then, Equation (III.5) can be rewritten as

$$\begin{aligned} \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G},A} &= P_{1,0} - P_{L,n_L} \\ &= \sum_{l=1}^{L-1} P_{l,0} - P_{l,n_l} \ . \end{aligned} \tag{III.6}$$

Our goal is to show that

$$P_{l,0} - P_{l,n_l} \leq n_l \ \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{G}_l}(a_l, t) \tag{III.7}$$

for any $l \in \{1, \ldots, L-1\}$. The claimed result would then follow directly from this since A is an arbitrary adversary running in time t. We can do so by using the advantage function of each $\mathcal{G}_l$. Consider the following sequence of distinguishers D_l for each $\mathcal{G}_l$.

```
algorithm D_l(S)   (|S| = a_l k_{l+1})      (1 ≤ l ≤ L − 1)
    j ←R {0, ..., n_l − 1} ; i ← 0 ; c_l ← 0
    parse S as S_0 ‖ ... ‖ S_{a_l − 1}
    if c_l < j then
        Out_h ←R {0,1}^{k_{h+1}}
    else if c_l = j then
        Out_h ← S_i ; i ← i + 1
    else
        Out_{h−1} ←R {0,1}^{k_h} ; St_h ← K_h(Out_{h−1})
        (Out_h, St_h) ← N_h(St_h)
    for h = l + 1, ..., L − 1 do
        St_h ← K_h(Out_{h−1})
        (Out_h, St_h) ← N_h(St_h)
    s ← Out_{L−1} ; c_L ← 0
    for c_L = 1, ..., L − 1 do
        d ← c_L ; h ← L − 1
        while d mod a_h = 0 and h ≥ l do
```

$d \leftarrow \lfloor d/a_h \rfloor$; $h \leftarrow h - 1$

`if` $h = l - 1$ `then`

$c_l \leftarrow c_l + 1$; $h \leftarrow h + 1$

`if` $c_l < j$ `then`

$Out_h \xleftarrow{R} \{0,1\}^{k_{h+1}}$

`else if` $c_l = j$ `then`

$Out_h \leftarrow S_i$; $i \leftarrow i + 1$

`else`

$Out_{h-1} \xleftarrow{R} \{0,1\}^{k_h}$; $St_h \leftarrow \mathcal{K}_h(Out_{h-1})$

$(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$

`else if` $h = l$ `and` $c_l < j$ `then`

$Out_h \xleftarrow{R} \{0,1\}^{k_{h+1}}$

`else if` $h = l$ `and` $c_l = j$ `then`

$Out_h \leftarrow S_i$; $i \leftarrow i + 1$

`else`

$(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$

`while` $h < L - 1$ `do`

$h \leftarrow h + 1$; $St_h \leftarrow \mathcal{K}_h(Out_{h-1})$

$(Out_h, St_h) \leftarrow \mathcal{N}_h(St_h)$

$s \leftarrow s \,\|\, Out_{L-1}$

$g \leftarrow A(\mathsf{guess}, s)$

`return` $1 - g$

Suppose we run experiment $\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G}_l,a_l,D_l}$. We notice it amounts to running experiment $\mathbf{Hybrid}_{A,l,j}$, where j is the value chosen at random by D_l in its first step, and then flipping the answer bit. Similarly if we run experiment $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G}_l,a_l,D_l}$ we notice that it amounts to running $\mathbf{Hybrid}_{A,l,j-1}$ where j is the value chosen at random by D_l in its first step, and then flipping the answer bit. So

$$\Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G}_l,a_l,D_l} = 1\,] \quad = \quad \tfrac{1}{n_l}\textstyle\sum_{j=1}^{n_l} 1 - P_{l,j}$$

$$\Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G}_l,a_l,D_l} = 1\,] \quad = \quad \tfrac{1}{n_l}\textstyle\sum_{j=1}^{n_l} 1 - P_{l,j-1}\ .$$

Subtract the second sum from the first and exploit the collapse to get

$$\frac{P_{l,0} - P_{l,n_l}}{n_l} \;=\; \tfrac{1}{n}\textstyle\sum_{j=1}^{n_l}(1 - P_{l,j}) \;-\; \tfrac{1}{n_l}\textstyle\sum_{j=1}^{n_l}(1 - P_{l,j-1}) \;=\; \mathbf{Adv}^{\text{prg}}_{\mathcal{G}_l,D_l}\ .$$

Note that the running time of D_l is at most t, whence we get Equation (III.7). This concludes the proof of the theorem. ∎

III.C.3 Discussion

One direct implication of the above theorem is that this scheme can generate up to n_L keys even though each base scheme $\mathcal{G}_l$ could only generate a_l keys; this is an added advantage of the construction. For example, let $n_L = Q$. Then the factor multiplying the advantage function of each $\mathcal{G}_{L-1}$ is Q/a_{L-1}. In this particular case, a trade-off between the factor Q/a_{L-1} and the advantage function of $\mathcal{G}_{L-1}$ over a_{L-1} queries will play the major role in the overall behavior of $\mathcal{G}$. On the one hand, if we choose a small value for a_{L-1}, opting for relaxing the constraints over $\mathcal{G}_{L-1}$, then we decrease the total number of data blocks Q generated by $\mathcal{G}$. This can be compensated, however, by increasing the values of other a_l. On the other hand, if we pick a large value for a_{L-1}, thus increasing Q, then we can increase the requirements over $\mathcal{G}_{L-1}$, which in turn may be affect the overall efficiency. The right choice of values will depend on the requirements of each application. Our goal here is to provide a tool to ease its design.

III.C.4 Optimality of analysis

Theorem III.C.1 gives an upper bound for the advantage function of $\mathcal{G}$ in terms of the advantage function of each $\mathcal{G}_l$. However, it is not clear whether this upper bound is tight or whether one can do better. In this section, we show that the given analysis is tight and that one cannot do better without resorting to the peculiarities of a particular generator. In order to prove our claim, we follow the same line presented in [7]. That is, we first define a setting in which functions can

only be accessed as a black box and then present a simple algorithm in this setting which achieves a distinguishing probability equal (up to a constant factor) to that one given in the theorem.

We start by defining the new setting, BBF, standing for Black-Box-Family setting. In the BBF setting, for each data block length k, there exists a family $F^k : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$ of functions which takes as input a seed s and a value x and outputs some value y, all of length k. Each seed s determines a different function F_s^k. Since we want these functions to be accessed as a black box, no explicit description of how to compute them is given. Instead, the data blocks outputted by such functions can only be obtained by means of oracle queries. There are two types of queries. One is called *oracle query* and is related to a particular function being analyzed. The second type is called *family query* and can be related to any function in the family. Consequently, the seed has to be provided explicitly in the query. Each generator $\mathcal{G}^{F^k(\cdot,\cdot)} = (\mathcal{K}^{F^k(\cdot,\cdot)}, \mathcal{N}^{F^k(\cdot,\cdot)})$, whose data block length is k, is defined as follows in the BBF setting. $\mathcal{K}^{F^k(\cdot,\cdot)}$ takes as input a seed s and returns a tuple $\langle s, 0\rangle$. $\mathcal{N}^{F^k(\cdot,\cdot)}$ takes as input a tuple $\langle s, i\rangle$ and returns a pair $(F_s^k(i), \langle s, i+1\rangle)$. A distinguisher for a generator $\mathcal{G}^{F^k}$ in the BBF setting is an algorithm for deciding whether a given string is random or whether it is the output of this generator for some random seed s.

Let us proceed with the analysis. Assume, for the sake of simplicity, that, in our tree-based construction, all data blocks have length k (i.e., $k_l = k$ for $l = 1, \ldots, L$). Moreover, also assume the arity to be the same in all levels and equal to a. Let us then define $\mathbf{Adv}^{\mathrm{prg}}_{F^k,\mathcal{G}^{F^k}}(q,\tau)$ as being the maximum over all distinguishing probabilities in breaking the security of a generator $\mathcal{G}^{F^k}$ by any algorithm which makes at most q oracle queries and τ family queries with respect to a family F^k. Let $F^\star$ be the family obtained by using our tree-based construction. Then, for any family F^k, Theorem III.C.1 implies that

$$\mathbf{Adv}^{\mathrm{prg}}_{F^\star,\mathcal{G}}(n_L,\tau) \leq \sum_{l=1}^{L-1} n_l \; \mathbf{Adv}^{\mathrm{prg}}_{F^k,\mathcal{G}^{F^k}}(a,\tau) \;, \tag{III.8}$$

where $n_l = a^{l-1}$. Our goal is to show that such general upper bound is tight by providing a lower bound for functions drawn from a particular family.

Consider the family RF in which each of the 2^k functions with data block length k is a random map between $\{0,1\}^k$ and $\{0,1\}^k$. We claim that

$$\mathbf{Adv}^{\mathrm{prg}}_{RF^\star,\mathcal{G}}(n_L, O(n_L)) \geq c\ n_{L-1}\ \mathbf{Adv}^{\mathrm{prg}}_{RF^k,\mathcal{G}^{RF^k}}(a, O(n_L))\ , \tag{III.9}$$

for some $c > 0$. But from inequality III.8, we now that

$$\mathbf{Adv}^{\mathrm{prg}}_{RF^\star,\mathcal{G}}(n_L, O(n_L)) \leq c'\ n_{L-1}\mathbf{Adv}^{\mathrm{prg}}_{RF,\mathcal{G}}(a, O(n_L)) \tag{III.10}$$

for some constant $c' > 0$. (Theorem III.C.1 actually contains other low-order exponents terms, but we disregard them here for simplicity, although a distinguisher which matches even those terms can be easily constructed.) Putting both inequalities III.9 and III.10 together concludes our proof.

In order to prove inequality III.9, we shall proceed in steps. First, we construct a general distinguisher for the overall scheme and compute its advantage $\mathbf{Adv}^{\mathrm{prg}}_{RF^\star,\mathcal{G}}(n_L, \tau)$. Then, we relate this advantage to that of a single function, $\mathbf{Adv}^{\mathrm{prg}}_{RF,\mathcal{G}^{RF^k}}(a, \tau)$, and conclude the proof.

Our construction of the distinguishing algorithm exploits the fact that collisions in the tree-based construction can be detected at the output. For instance, if two output blocks at the same level are the same, then both subtrees rooted at these output blocks will also be the same and this can be easily detected solely by looking at the output string. However, collisions between data blocks at different levels can also be exploited. For example, if there is a collision between a data block at the last level and one at the level before that, we can find that out by using each of the data blocks in the output string as a seed to a generator and checking whether a group of a data blocks matches. The resulting advantage of such distinguisher is $O(n_L \cdot n_{L-1})/2^k$ (this is the probability of having a collision between these two levels) but it would require $O(a \cdot n_L)$ many queries. In order to lower the total number of queries, our construction only searches for matches of a fixed number of data blocks, say 3, instead of a. The resulting advantage,

though smaller, is still $O(n_L \cdot n_{L-1})/2^k$ while only $O(n_L)$ number of queries is now required. Thus,

$$\mathbf{Adv}^{\mathrm{prg}}_{RF^\star,\mathcal{G}}(n_L, O(n_L)) \geq \frac{c' \cdot n_L \cdot n_{L-1}}{2^k} \tag{III.11}$$

for some constant $c' > 0$. But it is easy to see that

$$\mathbf{Adv}^{\mathrm{prg}}_{RF,\mathcal{G}^{RF^k}}(a, \tau) \leq \frac{\tilde{c} \cdot \tau}{2^k} \ , \tag{III.12}$$

for some constant $\tilde{c} >)0$. Therefore,

$$\mathbf{Adv}^{\mathrm{prg}}_{RF^\star,\mathcal{G}}(n_L, O(n_L)) \geq c \cdot n_{L-1} \cdot \mathbf{Adv}^{\mathrm{prg}}_{RF,\mathcal{G}^{RF^k}}(a, O(n_L)) \ , \tag{III.13}$$

for some constant $c > 0$, which concludes our proof. It is worth to say that an even tighter result can be obtained by making the distinguisher check for collisions between any two levels and within each level. Although this would result in a larger advantage, the order of total number of queries is still kept the same.

III.C.5 More general constructions

Even though the tree-based construction presented above allows us for some flexibility when instantiating the scheme, it requires all generators in the same level to be the same. This seems to be too strict. Hybrid schemes using different types of generators in the same level may be desirable in some situations since security requirements may vary with time (based on statistics, for example). We claim here that this restriction can be relaxed without compromising the security of the overall generator. In fact, we claim that the advantage function of the overall generator would simply be, in this case, the sum of the advantage function of each generator (represented by internal nodes in the tree). The proof for this claim would also be very similar to that for Theorem III.C.1 and so we skip it here.

Another possible generalization of the above scheme would involve the use of unbalanced trees. Though not so clearly, this case is actually covered by the construction just described above. Since each internal node can represent a different generator, it might as well be another tree-based construction. Consequently, such construction inherits the security of its building blocks.

III.D Re-keyed symmetric encryption

We fix a *base encryption scheme*. (For example, CBC mode encryption based on some block cipher.) We wish to encrypt data using this scheme, but with re-keying. Two things need to be decided. The first is how the re-keying is to be done, meaning how the subkeys will be computed. This corresponds to making a choice of stateful generator to generate the subkey sequence. The second is the lifetime of each subkey, meaning how many encryptions will be done with it. This corresponds to choosing an integer parameter $l > 0$ which we call the *subkey lifetime*. Associated to a base scheme, generator and subkey lifetime, is a particular *re-keyed encryption scheme*. We are interested in comparing the security of the re-keyed encryption scheme across different choices of re-keying processes (i.e. generators), keeping the base scheme and subkey lifetime fixed. In particular, we want to compare the use of the parallel and serial generators.

Our analysis takes a modular approach. Rather than analyzing separately the re-keyed encryption schemes corresponding to different choices of generators, we first analyze the security of a re-keyed encryption scheme with an arbitrary generator, showing how the advantage of the encryption scheme can be bounded in terms of that of the generator and the base scheme. We then build on results of Section III.B to get results for re-keyed encryption with specific generators. We begin by specifying in more detail the re-keyed encryption scheme and saying how we measure security of symmetric encryption schemes.

RE-KEYED ENCRYPTION SCHEMES. Let $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be the base (symmetric) encryption scheme, specified by its key generation, encryption and decryption algorithms [8]. Let $\mathcal{G} = (\mathcal{K}_g, \mathcal{N})$ be a stateful generator with block size k, where k is the length of the key of the base scheme. Let $l > 0$ be a subkey lifetime parameter. We associate to them a *re-keyed encryption scheme* $\overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{G}, l] = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$. This is a stateful encryption scheme which works as follows. The initial state of the encryption scheme includes the initial state of

the generator, given by $St_0 \stackrel{R}{\leftarrow} \mathcal{K}_g$. Encryption is divided into stages $i = 1, 2, \ldots$. Stage i begins with the generation of a new key K_i using the generator: $(K_i, St_i) \leftarrow \mathcal{N}(St_{i-1})$. In stage i encryption is done using the encryption algorithm of the base scheme with key K_i. An *encryption counter* is maintained, and when l encryptions have been performed, this stage ends. The encryption counter is then reset, the stage counter is incremented, and the key for the next stage is generated. If the base scheme is stateful, its state is reset whenever the key changes.

Formally, the key generation algorithm $\overline{\mathcal{K}}$ of the re-keyed scheme is run once, at the beginning, to produce an initial state which is shared between sender and receiver and includes St_0. The encryption algorithm $\overline{\mathcal{E}}$ takes the current state (which includes K_i, St_i, a stage counter, the encryption counter, and a state for the base scheme if the latter happens to be stateful) and the message M to be encrypted, and returns ciphertext $C \leftarrow \mathcal{E}_{K_i}(M)$. It also returns an updated state which is stored locally. It is advisable to include with the ciphertext the number i of the current stage, so that the receiver can maintain decryption capability even if messages are lost in transit. The $\overline{\mathcal{D}}$ algorithm run by the receiver can be stateless in this case. (This is true as long as the goal is privacy against chosen-plaintext attacks as we consider here, but if active attacks are considered, meaning we want privacy against chosen-ciphertext attacks or authenticity, the receiver will have to maintain state as well.)

Security measures for encryption schemes. Several (polynomial-time equivalent) definitions for security of a symmetric encryption scheme under chosen-plaintext attack were given in [8]. We use one of them, called left-or-right security. The game begins with a random bit b being chosen. The adversary then gets access to an oracle which can take as input any two equal-length messages (x_0, x_1) and responds with a ciphertext formed by encrypting x_b. The adversary wins if it can eventually guess b correctly. We can associate to any adversary an advantage measuring the probability it wins. We then associate to the base encryption scheme —respectively, the re-keyed encryption scheme— an advan-

tage function $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t,q,m)$ —respectively $\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t,q,m)$— which measures the maximum probability of the scheme being compromised by an adversary running in time t and allowed q oracle queries each consisting of a pair of m-bit messages. Intuitively, this captures security against a chosen-plaintext attack of q messages. (The usual convention [8] is to allow messages of different lengths and count the sum of the lengths of all messages but for simplicity we ask here that all messages have the same length. Note that for the base encryption scheme, all encryption is done using a single, random key. For the re-keyed scheme, it is done as the scheme specifies, meaning with the key changing every l encryptions. We omit details here, but precise definitions with this type of notation can be found for example in [14].)

Security of re-keyed encryption. The qualitative interpretation of the following theorem is that if the generator and base encryption scheme are secure then so is the re-keyed encryption scheme. It is the quantitative implications however on which we focus. The theorem says that the security of encrypting ln messages with the re-keyed scheme relates to the pseudorandomness of n blocks of the generator output and the security of encrypting l messages under the base scheme with a single random key. The $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t,l,m)$ term is multiplied by n, yet there is a clear gain, in that the security of the base encryption scheme relates to encrypting only l messages.

Theorem III.D.1 (Security of re-keyed encryption) Let $\mathcal{SE}$ be a base encryption scheme with key size k, let $\mathcal{G}$ be a stateful generator with blocksize k, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE},\mathcal{G},l]$ be the associated re-keyed encryption scheme. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t,ln,m) \;\le\; \mathbf{Adv}^{\text{prg}}_{\mathcal{G},n}(t) + n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t,l,m)\ . \blacksquare$$

Proof: Let $\overline{A}$ be an adversary attacking the security of the $\overline{\mathcal{SE}}$ scheme and let t be the maximum of the running times of experiments $\mathbf{Exp}^{\text{ind-cpa-}b}_{\overline{\mathcal{SE}},\overline{A}}$, where $b \in \{0,1\}$. We want to upper bound $\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}},\overline{A}} = \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\overline{\mathcal{SE}},\overline{A}} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\overline{\mathcal{SE}},\overline{A}} = 1\,]$.

To do this we specify an adversary D attacking the security of the stateful generator $\mathcal{G}$, and a distinguisher A attacking the $\mathcal{SE}$ scheme, and then bound the advantage of $\overline{A}$ in terms of the advantages of A and D.

THE ADVERSARY D. D will receive a sequence of blocks and must tell whether they are outputs of the generator or truly random. It will let the blocks it receives play the role of the keys k_i that are used by $\mathcal{SE}$ in the $\overline{\mathcal{SE}}$ scheme. D will test whether or not $\overline{A}$ succeeds on the given sequence of blocks. If so, it bets that the block sequence was pseudorandom, and if not, it bets that the block sequence was random. In adopting the latter opinion, it is assuming that breaking $\overline{\mathcal{SE}}$ is hard on a random block sequence, which we bear out later by providing a distinguisher which breaks the given (standard) encryption scheme $\mathcal{SE}$ otherwise. The description of D is as follows:

```
algorithm D(s)
    parse s as Out_1 || ... || Out_n
    b <-R {0,1}
    for i = 1,...,n do
        St <- A^{<E(Out_i, LR(b',·,·)), i>}(St)
    g <- A(St)
    if g = b then return 1 else return 0
```

At stage i, it uses block Out_i to simulate the encryption oracle $\overline{\mathcal{E}}(K_i, \mathrm{LR}(b, \cdot, \cdot))$ that $\overline{A}$ is supposed to get at that stage, given the definition of $\overline{\mathcal{SE}}$. The notation $\overline{A}^{\langle \mathcal{E}(Out, \mathrm{LR}(b,\cdot,\cdot)), i\rangle}$ means that $\overline{A}$ is given an oracle that on input (x_0, x_1) returns $\langle \mathcal{E}(Out_i, x_b), i\rangle$. When $\overline{A}$ outputs its guess, A checks whether $\overline{A}$ makes the right guess. If so, then it returns 1, betting that the output is not random, else 0.

We notice that the simulation of encryption oracle in each stage is perfect when the output blocks come from the generator. Hence, we have that

$$\Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,D} = 1\,] \;=\; \mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}},\overline{A}}\;. \tag{III.14}$$

THE DISTINGUISHER A. We design a distinguishing algorithm A attacking the given scheme $\mathcal{SE}$. A gets an oracle $\mathcal{E}(K, \mathrm{LR}(b', \cdot, \cdot))$. It runs $\overline{A}$, but on a sequence of random, independent keys rather than keys obtained via the generator. It also simulates the LR encryption oracle $\langle \mathcal{E}(\mathit{Out}, \mathrm{LR}(b, \cdot, \cdot)), i \rangle$ that should be given to $\overline{A}$. However, it does so differently from what $\overline{A}$ expects. That is, instead of selecting the bit b at random and using it in simulation of the LR encryption oracle throughout all time periods, it picks a period $j \in \{1, \ldots, n\}$ at random and sets the value b, used in the simulation of the LR encryption oracle, to 0 for all time periods $i < j$ and to 1 for all time periods $i > j$. In time period j, it lets K play the role of K_j by using its own oracle $\mathcal{E}(K, \mathrm{LR}(b', \cdot, \cdot))$ in the simulation of $\langle \mathcal{E}(\mathit{Out}, \mathrm{LR}(b, \cdot, \cdot)), j \rangle$. The intuition for doing so is that in order for $\overline{A}$ to be able to distinguish between the left or right encryption throughout all time periods, it should also be able to identify the case in which we start by encrypting the left input to the LR oracle and then switch to encrypting the right input to the LR oracle. At the end, A outputs the same guess $\overline{A}$ does. In running $\overline{A}$, A also keeps track of the state St of $\overline{A}$ which is passed from the current stage to the next.

```
algorithm A^{E(K,LR(b',·,·))}
    j <-R {1,...,n} ; s <- ε
    for i = 1,...,n do
        K_i <-R {0,1}^k
        if  i < j then  b <- 0
        if  i > j then b <- 1
        if i = j then
            St <- Ā^{⟨E(K,LR(b',·,·)),i⟩}(St)
        else
            St <- Ā^{⟨E(K_i,LR(b,·,·)),i⟩}(St)
    g <- Ā(St)
    return  g
```

In order to analyze the success probability of A, let us define the following sequence of hybrid experiments, where j varies between 0 and n.

```
experiment Hybrid_{Ā,j}
    for i = 1, ..., n do
        K_i <-R {0,1}^k
        if i ≤ j then b ← 0 else b ← 1
        St ← Ā^{⟨E(K_i, LR(b,·,·)),i⟩}(St)
    g ← Ā(St)
    return g
```

Suppose that $b' = 0$ in the oracle given to A. We can see that, in this case, the probability that A returns 1 equals the probability that the experiment $\mathbf{Hybrid}_{\overline{A},j-1}$ returns 1, where j is the value chosen at random by A in its first step. Similarly, if $b' = 1$ in the oracle given to A, then the probability that A returns 1 equals the probability that the experiment $\mathbf{Hybrid}_{\overline{A},j}$ returns 1, where j is the value chosen at random by A in its first step. Let P_j be the probability that experiment $\mathbf{Hybrid}_{\overline{A},j}$ returns 1, for $j = 0, \ldots, n$. Then,

$$\begin{aligned} \mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE},A} &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \\ \mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE},A} &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j} \ . \end{aligned}$$

If we subtract the second sum from the first and exploit the collapse, we get that

$$\begin{aligned} \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},A} &= \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE},A} = 1\,] \\ &= \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j-1} \ - \ \tfrac{1}{n}\textstyle\sum_{j=1}^{n} P_{j} \\ &= \frac{P_n - P_0}{n} \ . \end{aligned}$$

Moreover, by noticing that $P_n - P_0$ equals the probability that $\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,D}$ returns 1, we have that

$$\Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,D} = 1\,] \ = \ n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},A} \tag{III.15}$$

Now, combining Equations (III.14) and (III.15), we get

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}},A} &= \Pr[\,\mathbf{Exp}^{\text{prg-real}}_{\mathcal{G},n,b,A} = 1\,] \\ &= \mathbf{Adv}^{\text{prg}}_{\mathcal{G},A} + \Pr[\,\mathbf{Exp}^{\text{prg-rand}}_{\mathcal{G},n,b,A} = 1\,] \\ &= \mathbf{Adv}^{\text{prg}}_{\mathcal{G},A} + n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},A}\ .\end{aligned}$$

Finally, by observing that the running time of both D and A is at most t and that D makes at most q queries to its encryption oracle, it follows that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(qn,t) \;\le\; \mathbf{Adv}^{\text{prg}}_{\mathcal{G}}(n,t_1) + n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(q,t_2)\ .$$

∎

Re-keyed encryption with the parallel and serial generators. Combining Theorem III.D.1 with Theorems III.B.4 and III.B.5 gives us information about the security of re-keyed encryption under the parallel and serial generators.

Corollary III.D.2 (Security of re-keyed encryption with the parallel generator) Let $\mathcal{SE}$ be a base encryption scheme, let $F\colon\ \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF, let $\mathcal{PG}[F]$ be the F-based parallel generator defined in Construction III.B.1, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{PG}[F], l]$ be the associated re-keyed encryption scheme. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t,ln,m) \;\le\; \mathbf{Adv}^{\text{prf}}_{F}(t,n) + n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t,l,m)\ .\blacksquare$$

Corollary III.D.3 (Security of re-keyed encryption with the serial generator) Let $\mathcal{SE}$ be a base encryption scheme, let $F\colon\ \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a PRF, let $\mathcal{SG}[F]$ be the F-based serial generator defined in Construction III.B.2, and let $l > 0$ be a subkey lifetime. Let $\overline{\mathcal{SE}} = \overline{\mathcal{SE}}[\mathcal{SE}, \mathcal{SG}[F], l]$ be the associated re-keyed encryption scheme. Then

$$\mathbf{Adv}^{\text{ind-cpa}}_{\overline{\mathcal{SE}}}(t,ln,m) \;\le\; n \cdot \mathbf{Adv}^{\text{prf}}_{F}(t+\log n,2) + n \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(t,l,m)\ .\blacksquare$$

Example. For the base encryption scheme, let us use CBC with some block cipher $B\colon\ \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^b$ having block length b. We wish to

compare the security of encrypting q messages directly with one key; doing this with re-keying using the parallel generator; and doing this with re-keying using the serial generator. The re-keying is based on a PRF F: $\{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ having block length k. Note that B and F can but need not be the same. In particular B must be a cipher (i.e. invertible) in order to enable CBC decryption, but we have seen that better security results for the re-keying schemes by choosing F to be non-invertible and might want to choose F accordingly.

Let $\mathcal{CBC}$ denote the base encryption scheme. Let $\mathcal{PCBC}$ denote the re-keyed encryption scheme using $\mathcal{CBC}$ as the base scheme, the F-based parallel generator, and subkey lifetime parameter l. Let $\mathcal{SCBC}$ denote the re-keyed encryption scheme using $\mathcal{CBC}$ as the base scheme, the F-based serial generator, and subkey lifetime parameter l. Since B is a cipher we take its advantage to be

$$\mathbf{Adv}^{\text{prf}}_{B}(t,q) \approx \frac{q^2}{2^b} + \frac{t}{2^k} \ . \tag{III.16}$$

We know from [8] that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t,q,m) \approx \frac{q^2m^2}{b^2 2^b} + 2 \cdot \mathbf{Adv}^{\text{prf}}_{B}(t,qm/b) \approx \frac{3q^2m^2}{b^2 2^b} + \frac{2t}{2^k} \ .$$

For simplicity we let the message length be $m = b$. Thus if $q = ln$ messages of length m are CBC encrypted we have

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t,ln,b) &\approx \frac{3l^2n^2}{2^b} + \frac{2t}{2^k} \\
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PCBC}}(t,ln,b) &\approx \mathbf{Adv}^{\text{prf}}_{F}(t,n) + \frac{3l^2n}{2^b} + \frac{2nt}{2^k} \\
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SCBC}}(t,ln,b) &\approx n \cdot \mathbf{Adv}^{\text{prf}}_{F}(t+\log n,2) + \frac{3l^2n}{2^b} + \frac{2nt}{2^k} \ .
\end{aligned}$$

The first corresponds to encryption with a single key, the second to re-keying with the parallel generator, and the third to re-keying with the serial generator. Suppose we let F be a block cipher. (This is the easiest choice in practice.) We can simply let $F = B$. In that case F obeys Equation (III.3) and we get

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{CBC}}(t,ln,b) &\approx \frac{3l^2n^2 + 2t}{2^k} \\
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PCBC}}(t,ln,b) &\approx \frac{3l^2n + n^2 + 2nt}{2^k}
\end{aligned}$$

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SCBC}}(t, ln, b) \approx \frac{3l^2n + 2nt + t}{2^k} .$$

The two generators deliver about the same advantage. To gauge the gains provided by the re-keying schemes over the single-key scheme, let us define the *encryption threshold* of a scheme to be the smallest number of messages $Q = ln$ that can be encrypted before the advantage hits one. (Roughly speaking, this is the number of messages we can safely encrypt.) We want it to be as high as possible. Let's take $t \approx nl$. (It cannot be less but could be more so this is an optimistic choice). In the single-key scheme $Q \approx 2^{k/2}$. In the re-keyed schemes let us set $l = 2^{k/3}$. (This is the optimal choice.) In that case $Q \approx 2^{2k/3}$. This is a significant increase in the encryption threshold, showing that re-keying brings important security benefits.

We could try to set F to be a non-invertible pseudorandom function for which Equation (III.4) is true. (In particular F would not be B.) Going through the calculations shows that again the two generators will offer the same advantage, but this would be an improvement over the single-key scheme only if $k > b$. (Setting $k = 2b$ yields an encryption threshold of 2^b for the re-keyed schemes as compared to $2^{b/2}$ for the single-key scheme.)

We saw in Section III.B that the parallel generator offered greater security than the serial one. We note that this did not materialize in the application to re-keyed CBC encryption: here, the advantage functions arising from re-keying under the two generators are the same. This is because the term corresponding to the security of the base scheme in Corollaries III.D.2 and III.D.3 dominates when the base scheme is CBC.

In summary we wish to stress two things: that security increases are possible, and that our results provide general tools to estimate security in a variety of re-keyed schemes and to choose parameters to minimize the advantage functions of the re-keyed schemes.

Chapter IV

Broadcast Encryption Schemes

IV.A Introduction

The domain we consider in this chapter is that of broadcast applications where the transmissions need to be encrypted. The examples we consider are a broadband digital TV network [58], broadcasting either via satellite or via cable, and Internet secure multicast [78], e.g., via the MBone [36].

In the context of **pay TV**, the *head-end* occasionally needs to multicast an encrypted message to some subset of users (called the target set) using the broadcast channel. Each network user has a *set-top terminal* (STT) which receives the encrypted broadcast and decrypts the message, if the user is entitled to it. For this purpose the STT securely stores the user's secret keys, which we refer to as establishment keys. Because of extensive piracy [60], the STTs need to contain a secure chip which includes secure memory for key storage. This memory should be non-volatile, and tamper-resistant, so the pirates will find it difficult to read its contents. As a result of these requirements, STTs have severely limited secure memory, typically in the range of a few Kilobytes.

Earlier work on broadcast encryption (cf. [38]) was motivated by the need to transmit the key for the next billing period or the key for the next pay-per-view event, in-band with the broadcast, since STTs only had uni-directional

communications capabilities. The implicit assumption was that users sign up for various services using a separate channel, such as by calling the service provider over the phone. In such applications it is reasonable to assume that the target set is almost all the population, and there are only small number of excluded users. Moreover, it is crucial that users outside the target set are not able to decrypt the message since it has a high monetary value, e.g., the cost of a month's subscription.

However, current STTs typically follow designs such as [28] which allow bi-directional communication, where the uplink uses an internal modem and a phone line, or a cable modem. These new STTs upload the users' requests and download next month's keys via a callback mechanism, and not through the broadcast channel. This technological trend would seem to invalidate the necessity for broadcast encryption schemes completely. We argue that this is not the case—there are other applications where broadcast encryption is necessary, such as multicasting electronic coupons, promotional material, and low-cost pay-per-view events. Such applications need to multicast short-lived, low value messages that are not worth the overhead of communicating with each user individually. In such applications, though, the requirements from the solution are slightly different. On the one hand, it is no longer crucial that only users in the target set receive the message, as long as the number of free-riders is controlled. On the other hand, it is no longer reasonable to assume anything about the size of the target set.

Multicast in the **Internet** is a service that is bound to become more and more popular. Audio and video are two most talked about applications, but there are diverse data applications that can benefit from multicast, such as news updates, stock quotes, etc. Some of the multicast applications will be broadcasted freely, while others will be for pay. Once such pay-multicast services are deployed on the Internet, they would face similar issues faced by the pay-TV industry: High-value transmissions would need to be encrypted, and only paying customers should have the decryption keys.

In the past years, several suggestions for Internet multicast key-manage-

ment architectures were proposed [63, 79, 78]. These proposals do not specify whether the decryption keys are to be held in a tamper-resistant hardware module or in the receiving hosts' insecure memory. We speculate that if Internet multicasts are to carry information with significant monetary value, then the service providers will encounter piracy. And in the Internet, the service providers loose the main advantage they have over the pirates in the pay-TV industry: The pirates have access to a high bandwidth, cheap, and world-wide, distribution channel—the same Internet that the service providers use (see [53] for a discussion of these issues). For this reason, we argue that successful Internet pay-multicast services would probably require users to keep keys in a secure STT-like device. But regardless of how Internet multicast keys are to be stored, it is certainly desirable to keep their number low.

For the rest of this chapter, we use the term *"receiver"*, for both pay-TV STTs and Internet-multicast key stores (implemented either in a secure module or in the host memory).

IV.A.1 Related Work

Fiat and Naor [38] were first to introduce broadcast encryption (in the context of pay-TV). They suggested methods of securely broadcasting key information such that only a selected set of users can decrypt this information while coalitions of up to k other users can learn nothing, either in the information-theoretic sense, or under a computational security model. Their schemes, though, required impractical numbers of keys to be stored in the receivers. Extensions to this basic work can be found in [22, 23, 77].

Luby and Staddon [57] studied the trade-off between the transmission length and the number of keys stored in the receivers. They assumed a security model in which encryptions cannot be broken, i.e., only users that have a correct key can decrypt the message. We adopt the same security model. Their work still addressed fixed-size target sets, which are assumed to be either very large or

very small, and no user outside the target set is allowed to be able to decrypt the message. A main part of their work is a disillusioning lower bound, showing that either the transmission will be very long or a prohibitive number of keys needs to be stored in the receivers.

A related line of work goes under the title of "tracing traitors" [27, 64]. The goal is to identify some of the users that leak their keys, once a cloned receiver is found. This is achieved by controlling which keys are stored in each receiver, in a way that the combination of keys in a cloned receiver would necessarily point to at least one traitor.

Key management schemes for encrypted broadcast networks, which give the vendor the flexibility to offer program packages of various sizes to the users, can be found in [80]. The problem of tracking the location of receivers in order to prevent customers from moving a receiver from, e.g., a home to a bar, is addressed in [40].

The Iolus project [63] was the first serious attempt to propose a framework for secure Internet multicast. The Iolus framework is based on a hierarchical tree structure. At higher layers, group servers communicate with each other using secure multicast. At lower layers, secure communication is exchanged between group members. Special servers are needed to handle the join/leave operation in each level.

Wong et al. [79] were the first to suggest a *key management* scheme for secure multicast in the Internet. Their work influenced the network working group of the IETF in the form of a recent RFC [78]. Their solution is based on a hierarchy of keys that is built with the group of currently paying customers. The customers of each service (a movie channel, a stock quote service, a news bulletin) are the leaves of a balanced tree, where each node of the tree corresponds to a group key; the group members are the descendents of that node. This solution achieves a logarithmic cost for join/leave operations. However, the tree is built per program (or per group), and as such incurs a high overhead when many different programs

are multicast.

IV.A.2 Contributions

Our starting point is the observation that the requirement "no users outside the target set can decrypt the message" is too strict for many applications. For instance, for the purposes of multicasting electronic coupons, it may be enough to guarantee that the recipient set contains the target set, and that the total number of recipients is no more than f times the size of the target set. Service providers can afford a small potential increase in the number of redeemed coupons, as long as this simplifies their operations and lowers their cost. We call establishment key allocation schemes that provide such guarantees "f-redundant broadcast encryption schemes". Relaxing the requirements in this way allows us to depart from the lower bounds of [57].

On the other hand, we have a more ambitious goal when it comes to possible target sets. Unlike earlier work, we require our schemes to be able to multicast to *any* target set, not just those target sets of very small or very large cardinality.

We concentrate on schemes which store only a small number of keys in each receiver. For systems with several million users, it is reasonable to require the maximum number of keys per user to be $O(\log n)$, where n is the total number of users, or at most $O(n^\epsilon)$, where, say, $\epsilon \leq 1/4$.

Subject to these constraints, we are interested in several measures of the quality of an establishment key allocation. The first is the number of transmissions, t: we can always attain our requirements trivially if we assign each receiver a unique key, but then we suffer a very high number of transmissions. The second parameter, which we call *opportunity*, is the proportion of free-riders in the population outside the target set. The opportunity measures the incentive a customer has to avoid paying (in cheap pay-per-view type services). If the opportunity is very high, close to 1, there is no incentive for customers to pay, as they can almost surely get a

free ride.

After discussing the basic trade-offs associated with the problem, we present some simple examples, that show the problem difficulty. We then prove a new lower bound on the tradeoff between the transmission length and the number of keys stored per receiver, a lower bound that incorporates the f-redundancy of our establishment key allocations. We show that the f-redundancy gives us a substantial gain: For the same number of transmissions t we can hope for only $\exp(\Omega(n/tf))$ keys per receiver, whereas the bound of [57] is $\exp(\Omega(n/t))$.

We then present several establishment key allocation constructions, and an approximation algorithm that finds a key cover with minimal number of transmissions, for any given target set of users. Since this problem is similar to the minimum set cover problem, that is known to be NP-hard, we cannot expect to find an optimal solution efficiently. Instead, we use a greedy approximation algorithm to find good key covers. We conducted an extensive simulation study of the problem, from which we present only the interesting results.

Finally, we discuss the practical aspects of using our scheme for key management of secure multicast on the Internet. We propose a single key management structure for all the services in a given infrastructure (e.g., one key structure for the entire MBone [36]). The cost of building this key management structure is logarithmic in the size of the total user population, but it is now usable for all the services provided on this infrastructure, making it much cheaper than the "separate tree per group" proposed by [79, 78]. We discuss how to build and maintain our structure incrementally, as users are added or dropped from the MBone. We also discuss the practical issues of how to manage the key transmissions in a dynamic environment, where paying users join and leave a specific program while it is in progress. We show that such a dynamic environment further encourages users to pay.

Our results are based on previous joint work with Yuval Shavitt and Avishai Wool [5, 4].

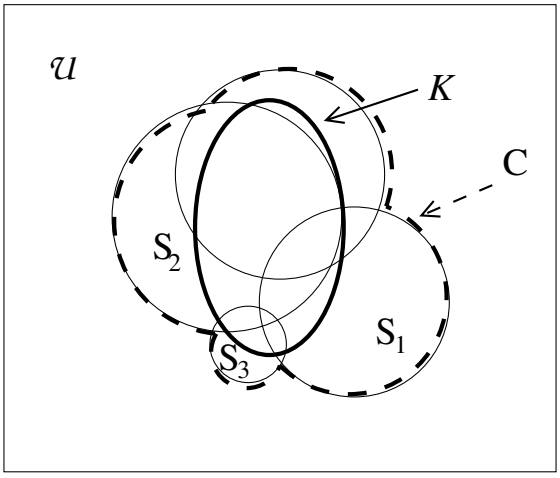

Figure IV.1: Definition depiction.

Organization: In the next section we formally define the problem and the various parameters we are interested in. In Section IV.C we show some simple solutions. In Section IV.D we prove our new lower bound on the trade-off between the number of keys per user, the redundancy factor, and the transmission length. In Section IV.E we discuss how to find which keys to use given an establishment key allocation. In Section IV.F we show our schemes and the results of their performance evaluation. In Section IV.G we discuss how to adapt our schemes to dynamic Internet environments, and evaluate their performance in such environments.

IV.B Definitions and Model

Let $\mathcal{U}$ be the set of all the receivers (i.e., receivers connected to a head-end), with $|\mathcal{U}| = n$. We use K to denote the *target set*, i.e., the set of paying customers, and denote its size by $|K| = k$ (see Figure IV.1).

We describe the allocation of the establishment keys by a collection $\mathcal{S} = \{S_1, S_2, \ldots\}$ of *key sets* such that $\cup S_i = \mathcal{U}$. We associate a unique establishment key e_i with each set $S_i \in \mathcal{S}$. A key e_i is stored in the secure memory of every receiver $u \in S_i$. Hence the number of keys a receiver $u \in \mathcal{U}$ stores is equal to the

number of sets $S_i \in \mathcal{S}$ it belongs to. Formally,

Definition IV.B.1 Let $\mathcal{S}$ be an establishment key allocation. The *degree* of a receiver u is $deg(u) = |\{i : S_i \ni u\}|$. The degree of a collection $\mathcal{S}$ is $deg(\mathcal{S}) = \max_{u \in \mathcal{U}} deg(u)$.

Definition IV.B.2 Given a target set K, a *key cover* of K is a collection of sets $S_i \in \mathcal{S}$ whose union contains K:

$$\mathcal{C}(K) \subseteq \mathcal{S} \text{ such that } K \subseteq \cup_{S_i \in \mathcal{C}(K)} S_i.$$

The *minimal key cover* is $\mathcal{C}_{\min}(K) = \mathcal{C}(K)$ for which $|\mathcal{C}(K)|$ is minimal.

Suppose the head-end needs to send a message μ to all the members of a target set K. Given any key cover $\mathcal{C}(K)$, the head end encrypts μ using the establishment keys e_i corresponding to the sets $S_i \in \mathcal{C}(K)$, and broadcasts each encryption separately.[1].

Definition IV.B.3 We denote the best possible number of transmissions that the head-end can use for a target set K by $t_K = |\mathcal{C}_{\min}(K)|$. Thus the worst case number of transmissions is $t_{\max}(\mathcal{S}) = \max_K \{ t_K \}$.

In order to define the redundancy and opportunity measures we need the following technical definition.

Definition IV.B.4 We denote the set of recipients of a given key cover $\mathcal{C}(K)$ by $R_{\mathcal{C}}(K) = \cup\{S_i \in \mathcal{C}(K)\}$ and the total number of recipients by $r_{\mathcal{C}}(K) = |R_{\mathcal{C}}(K)|$.

By the definition of a key cover $\mathcal{C}(K)$, every member of the target set K has at least one of the keys used to encrypt μ. However, other receivers outside K usually exist, which are also capable of decrypting the message. All our establishment key allocations are constructed with a worst case guarantee that there are never too many of these free-riders. Formally,

[1]This method was called the OR protocol in [57].

Definition IV.B.5 An establishment key allocation $\mathcal{S}$ is said to be *f-redundant* if

$$\frac{r_{\mathcal{C}}(K)}{k} \leq f$$

for every $K \subseteq \mathcal{U}$ with $|K| = k$.

A variant measure of redundancy is the *actual redundancy* f_a, which is the ratio between the non-paying and paying recipients. We are interested in the average case f_a, so we define it as a function of the target set K. Formally,

Definition IV.B.6 For a target set K with $|K| = k$ the *actual redundancy* is $f_a = \frac{r_{\mathcal{C}}(K)-k}{k}$.

If $\mathcal{S}$ guarantees a worst case redundancy factor f, then $0 \leq f_a \leq f - 1$ for any target set K.

Finally, we define the opportunity, η, as the proportion of non-paying recipients (free-riders) in the non-paying population ($0 \leq \eta \leq 1$). The opportunity measures the incentive a customer has to avoid paying (e.g., in cheap pay-per-view type services). Again, this is a function of the target set K.

Definition IV.B.7 For a target set K with $|K| = k$ the *opportunity* is $\eta = \frac{r_{\mathcal{C}}(K)-k}{n-k}$.

IV.C Simple Examples

To demonstrate our definitions and the trade-offs associated with the problem let us examine some simple solutions for the problem (which are similar to those in [79]). See Table IV.1 for a summary of the examples.

Example IV.C.1 The "always broadcast" solution: $\mathcal{S} = \{\mathcal{U}\}$.

Both the degree, $deg(\mathcal{S})$, and the number of transmissions, $t_{\max}(\mathcal{S})$, required to distribute the message are optimal and equal to 1 in this case. However, the redundancy is $f = n$ in the worst case and the opportunity, η, is always 1. The

$\mathcal{S}$	$deg(\mathcal{S})$	$t_{\max}(\mathcal{S})$	f	η
$\{\mathcal{U}\}$	**1**	**1**	n	1
$\{\{1\},\ldots,\{n\}\}$	**1**	n	**1**	**0**
$2^{\mathcal{U}}$	2^{n-1}	**1**	**1**	**0**

Table IV.1: A summary of some simple examples. Bold numerals indicate an optimal parameter.

last two parameters are very bad since the system gives no incentive for a customer to pay for a program; a single paying customer enables the entire population a free ride.

Example IV.C.2 The "key per user" solution: $\mathcal{S} = \{\{1\},\{2\},\ldots,\{n\}\}$.

Here the degree $deg(\mathcal{S}) = 1$ is optimal, and so are the redundancy $f = 1$, and the opportunity $\eta = 0$. However, the number of transmissions is a very poor $t_{\max}(\mathcal{S}) = n$.

Example IV.C.3 The "all possible sets" solution: $\mathcal{S} = 2^{\mathcal{U}}$.

The degree here is an impractical $deg(\mathcal{S}) = 2^{n-1}$, however, all the other parameters are optimal: $t_{\max}(\mathcal{S}) = 1$, $f = 1$, and $\eta = 0$. This is because every possible target set K has its own designated key.

IV.D A Lower Bound

IV.D.1 Tools

Before presenting our lower bound on the degree of an f-redundant establishment key allocation, we need to introduce some definitions and results which we use in the proof.

We start with *covering designs*, which are a class of combinatorial block designs. A succinct description of covering designs can be found in [29, Ch. IV.8]. A more detailed survey is [62].

Definition IV.D.1 A k-(n, d) *covering design* is a collection of d-sets *(blocks)* $\mathcal{D} = \{D_1, \ldots, D_\ell\}$ over a universe of n elements, such that every k-set of elements is contained in at least one block.

Definition IV.D.2 The *covering number* $C(n, d, k)$ is the minimum number of blocks in any k-(n, d) covering design.

Theorem IV.D.3 [Schönheim bound] [70] $C(n, d, k) \geq L(n, d, k)$, where

$$L(n, d, k) = \left\lceil \frac{n}{d} \left\lceil \frac{n-1}{d-1} \cdots \left\lceil \frac{n-k+1}{d-k+1} \right\rceil \right\rceil \right\rceil \geq \left(\begin{array}{c} n \\ k \end{array} \right) \Big/ \left(\begin{array}{c} d \\ k \end{array} \right).$$

We also rely on the following result of Luby and Staddon, which addresses *strict* broadcast encryption protocols.

Definition IV.D.4 An establishment key allocation $\mathcal{S}$ is called *strict* for a collection of target sets $\mathcal{D}$ if the sets in $\mathcal{D}$ can be covered without redundancy. Formally, $R_{\mathcal{C}}(D) = D$ for all $D \in \mathcal{D}$.

Theorem IV.D.5 [57] Let $\mathcal{D} = \{D_1, \ldots, D_\ell\}$ be a collection of target sets, with $|D_i| \geq d$ for all $D_i \in \mathcal{D}$. Then any establishment key allocation $\mathcal{S}$ which is strict for $\mathcal{D}$, and which can transmit to any $D_i \in \mathcal{D}$ using at most t transmissions, must have

$$deg(\mathcal{S}) \geq \left(\frac{\ell^{1/t}}{t} - 1 \right) \Big/ (n - d).$$

Remark: The precise statement we use here is a generalization of [57, Theorem 12]. In their original formulation the target sets D_i all have a cardinality of exactly d, and the collection $\mathcal{D}$ consists of all $\left(\begin{array}{c} n \\ d \end{array} \right)$ possible d-sets. However, their proof can be easily extended to any arbitrary collection of target sets, of cardinality d or larger.

IV.D.2 The Bound

Theorem IV.D.6 Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n, for which $t_{\max}(\mathcal{S}) = t$. Then

$$deg(\mathcal{S}) \geq \max_{1 \leq k \leq n/f} \left\{ \left(\frac{1}{t} \left[\binom{n}{k} \Big/ \binom{kf}{k} \right]^{1/t} - 1 \right) \Big/ (n-k) \right\}.$$

Proof: For a target set K of size $|K| = k$, let $R(K)$ be the minimal possible recipient set for K (or one such set if many minimal recipient sets exist). Consider the collection of minimal recipient sets

$$\mathcal{D} = \{R(K) : |K| = k\}.$$

Note that covering K f-redundantly, using the $t' \leq t$ key sets that define $R(K)$, is precisely equivalent to covering $R(K)$ *strictly* with (the same) t' key sets. Therefore we see that $\mathcal{S}$ is an establishment key allocation which is strict for $\mathcal{D}$, and can transmit to any $R(K) \in \mathcal{D}$ using at most t transmissions. Note also that, trivially, $|R(K)| \geq k$ for any $|K| = k$. Thus we can apply Theorem IV.D.5 to obtain

$$deg(\mathcal{S}) \geq \left(\frac{|\mathcal{D}|^{1/t}}{t} - 1 \right) \Big/ (n-k). \tag{IV.1}$$

By definition $|R(K)| \leq kf$ for all K, however, some sets $R(K) \in \mathcal{D}$ may have fewer than kf elements. Define a modified collection $\mathcal{D}'$ in which each $R(K) \in \mathcal{D}$ is replaced by some superset $\hat{R}(K) \supseteq R(K)$ with $|\hat{R}| = kf$. Note that $|\mathcal{D}'| \leq |\mathcal{D}|$ since $\hat{R}(K_1) = \hat{R}(K_2)$ is possible when $R(K_1) \neq R(K_2)$. But now $\mathcal{D}'$ is a k-(n, kf) covering design. Thus we can lower-bound its size by the Schönheim bound, Theorem IV.D.3, to obtain

$$|\mathcal{D}| \geq |\mathcal{D}'| \geq L(n, kf, k) \geq \binom{n}{k} \Big/ \binom{kf}{k}. \tag{IV.2}$$

Plugging (IV.2) into (IV.1) and maximizing the expression over the choice of k yields our result. ∎

Using standard estimations of binomial coefficients, and maximizing over k, we can obtain the following asymptotic estimate.

Corollary IV.D.7 Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n, for which $t_{\max}(\mathcal{S}) = t$. Then $deg(\mathcal{S}) \geq \exp(\Omega(n/tf))$. ∎

We therefore see that the f-redundancy gives us a substantial gain in the degree: the bound of [57] for strict establishment key allocations is $deg(\mathcal{S}) = \exp(\Omega(n/t))$. In other words, if we allow a redundancy factor of f we can hope to use only an f'th root of the number of keys required per receiver in a strict establishment key allocation for the same number of transmissions.

Theorem IV.D.6 and Corollary IV.D.7 give a lower bound on the required number of keys a receiver needs to store. As we said before, this is typically a small fixed value which we can reasonably model by $\log_2 n$ or n^ϵ. Thus we are more interested in the inverse lower bound, on the number of transmissions t. Asymptotically we can obtain the following bound.

Corollary IV.D.8 Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n. Then

$$t_{\max}(\mathcal{S}) \geq \begin{cases} \Omega\left(\frac{n}{f \log\log n}\right), & \text{when } deg(\mathcal{S}) = O(\log n), \\ \Omega(\frac{n}{f \log n}), & \text{when } deg(\mathcal{S}) = O(n^\epsilon). \end{cases}$$

The asymptotic bound of Corollary IV.D.8 hides the constants, and inverting Theorem IV.D.6 gives a rather unwieldy expression for the lower bound on t. Therefore, we choose to invert Theorem IV.D.6 numerically and to plot the result, as a function of the target set size k, in Figure IV.2. As we shall see in the sequel, the highest point on this curve ($t \approx 19$ for $n = 1024$) is significantly lower than our best constructions, which suffer from a worst case of $t_{\max}(\mathcal{S}) = 3n/8 = 384$ when $n = 1024$.

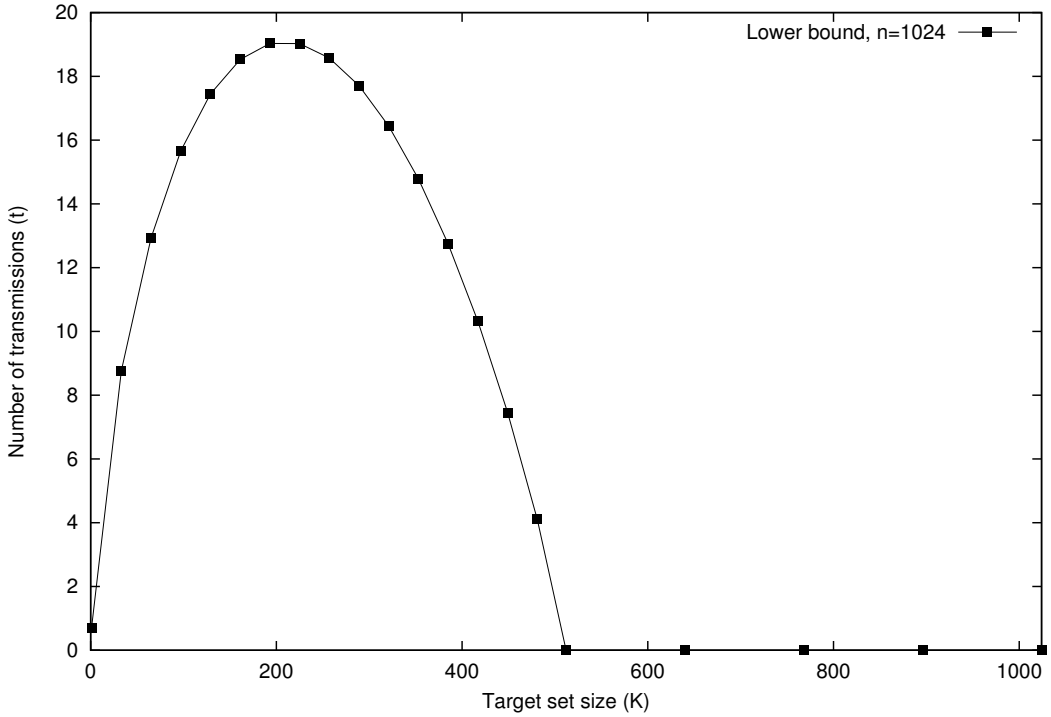

Figure IV.2: The lower bound for the number of transmissions (t) as a function of the target set size k, with $n = 1024$, $f = 2$, and $deg(\mathcal{S}) = \log_2 n$.

```
algorithm f-Cover (K, S)
  R ← ∅;   C ← ∅
  repeat
    A ← {S_i : |S_i \ R| / |(K ∩ S_i) \ R| ≤ f}.
    A ← S_i ∈ A which maximizes |(K ∩ S_i) \ R|.
    R ← R ∪ A;   C ← C ∪ {A}.
  until the candidate collection A is empty.
  return (R, C).
```

Figure IV.3: Algorithm *f-Cover*

IV.E Finding a Good Key Cover

An f-redundant establishment key allocation guarantees that an f-redundant cover exists for every target set K. In particular, singleton target sets $K = \{u\}$ need to be addressed. Thus, $\mathcal{S}$ must include enough sets S_i with $|S_i| \leq f$ so that every user is contained in one of them. For simplicity, we shall assume that $\mathcal{S}$ contains the singletons themselves as sets, i.e., every receiver is assumed to hold one key that is unique to it.

Once we decide upon a particular f-redundant establishment key allocation $\mathcal{S}$, we still need to show an efficient algorithm to find an f-redundant key cover $\mathcal{C}(K)$ for every target set K. Among all possible f-redundant key covers that $\mathcal{S}$ allows, we would like to pick the best one. By "best" we mean here a cover that minimizes the number of transmissions t. Trying to minimize the actual redundancy f_a would lead to trivialities because we can always achieve the optimal $f_a = 0$ since we assumed that $\mathcal{S}$ contains all the singletons. Thus, for every target set K, we obtain the following optimization problem: given a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ and a target set K, find a sub-collection $\mathcal{C}_{\min}(K) \subseteq \mathcal{S}$ with minimal cardinality $|\mathcal{C}_{\min}(K)|$ such that $K \subseteq \cup\{S_i \in \mathcal{C}_{\min}(K)\}$ and $|\cup\{S_i \in \mathcal{C}_{\min}(K)\}|/|K| \leq f$.

This is a variation of the Set Cover problem [41], and thus an NP-hard optimization problem. We omit the formal reduction proving this. Moreover, it

is known that no approximation algorithm exists for Set Cover with a worst case approximation ratio[2] better than $\ln n$ (unless NP has slightly super-polynomial time algorithms) [37].

On the positive side, the Set Cover problem admits a greedy algorithm, which achieves the best possible approximation ratio of $\ln n$ [52, 56]. Moreover, the greedy algorithm is extremely effective in practice, usually finding covers much closer to the optimum than its approximation ratio guarantees [45]. For this reason, our general algorithm *f-Cover* for choosing a key cover is an adaptation of the greedy algorithm. See Figure IV.3 for the details.

Theorem IV.E.1 If $\{\{1\}, \ldots, \{n\}\} \subseteq \mathcal{S}$ then algorithm *f-Cover* returns an f-redundant key cover of K for any target set K.

Proof: The set R maintains the current cover in the algorithm. In every iteration, when a set S_i is added to the cover, $|S_i \setminus R|$ new users are covered, and $|(K \cap S_i) \setminus R|$ of them are target set members that were not included in the cover before. Note that we only add a set S_i if $\frac{|S_i \setminus R|}{|(K \cap S_i) \setminus R|} \leq f$ and that the sets $(S_i \setminus R)$ are disjoint for the S_i's chosen in different iterations. From these observations it is easy to prove that the f-redundancy is kept throughout the algorithm execution, and in particular, when the algorithm terminates. ▮

Remarks:

- The candidate set $\mathcal{A}$ needs to be re-calculated in each iteration, since a non-candidate set S_i may become a candidate (or vice versa) after some other S_j is added to the cover.
- It is easy to see that the time complexity of algorithm *f-Cover* is $O(m^2)$ where m is the number of sets in $\mathcal{S}$.
- In practice the computation time on a 433 MHz Intel Celeron processor was between 1ms and 17ms for N =1024, 2048, and 4096, and various values of k.

[2] [49] contains a good discussion of approximation algorithms and in particular a chapter on Set Cover.

In order to make the algorithm even more efficient, we do not use it in its most general form. Instead, we split the establishment key allocation $\mathcal{S}$ into *levels*, each containing sets of the same size. Formally, we break $\mathcal{S}$ into $\mathcal{S} = \mathcal{S}^1 \cup \mathcal{S}^2 \cup \cdots$, such that $|S_i^\ell| = k_\ell$ for some k_ℓ and for all $S_i^\ell \in \mathcal{S}^\ell$. The algorithm is performed in phases, where only sets belonging to level $\mathcal{S}^\ell$ are considered in the candidate set $\mathcal{A}$ during in phase ℓ. The algorithm starts at the highest level, the one containing of the largest sets in $\mathcal{S}$. When $\mathcal{A}$ is empty at a certain level, the cover so far, R, and the covering sets, $\mathcal{C}$, are fed to the execution phase of the algorithm in the next (lower) level.

IV.F Practical Solutions

IV.F.1 Overview

Our basic goal is to construct an f-redundant establishment key allocation, namely to construct an $\mathcal{S}$ that will satisfy the following requirements: (i) the number of establishment keys per user (degree) is low; and (ii) $|R_{\mathcal{C}}(K)|/|K| \leq f$ for every target set $K \subseteq U$. Given such an establishment key allocation, we evaluate its performance with respect to the number of transmissions t, the actual redundancy f_a, and the opportunity η, using computer simulations.

We are interested in "average" performance, but since performance depends heavily on the target set size, k, we use it as a parameter in the simulations. Each data point for a target set size k in the graphs represents the mean of the relevant measure, averaged over r samples of k-sets chosen uniformly at random. We show the 95% confidence intervals[3] for each data point, unless the graphical height of the confidence intervals is very close to the size of the symbols depicted on the curves. We typically use $r = 25$ samples per data point.

Unless stated otherwise, we assume that the redundancy is $f = 2$. We also conducted experiments with other values of f but they showed qualitatively

[3] A 95% confidence interval means that the population mean appears within the specified interval with probability 0.95. See [54] for a precise definition of confidence interval.

similar results.

IV.F.2 The Tree Scheme

The Scheme's Description

A simple multi-level establishment key allocation is a balanced tree, that is built by recursively partitioning the sets of a high level into equally-sized, disjoint sets in the next level. Sets that form a partition of a single set, one level above them, are considered children of this set in the tree. The number of keys each receiver holds in this scheme is only $1 + \log_a n$, where a is the arity of the tree. In the sequel we always assume a binary tree ($a = 2$).

An important advantage of a tree scheme (besides its simplicity) is that the greedy algorithm of Figure IV.3 can easily be made to run in time linear in the size of the cover set, rather than in the total number of sets in the collection. The idea is to start at the root of the tree (the set $\mathcal{U}$) and then traverse it either in a DFS or in a BFS order. Whenever an f-redundant set is found, select it and ignore the subtree under it.

The problem with the tree scheme is its worst case behavior. Consider the case where $f = 2$ and the collection is a full binary tree. If the target set comprises $k = n/4$ users such that no two of them belong to a common set of size 4 or less, then we are forced to use $t = n/4$ transmissions. It is easy to see that this is the worst possible configuration.

The average behavior of the basic tree is substantially better than the worst case. Figure IV.4 shows the average number of transmissions on several variants of a tree for a population of $n = 1024$ users. We see from the "threshold at sets of size 2" curve in the figure that the peak of the average t is 164, which is 36% less than the worst case of 256. We explain this threshold and discuss the different variants of the tree in Section IV.F.2.

We conducted the same tests for larger populations and noticed that the qualitative behavior does not change significantly, thus we omit the details. Here

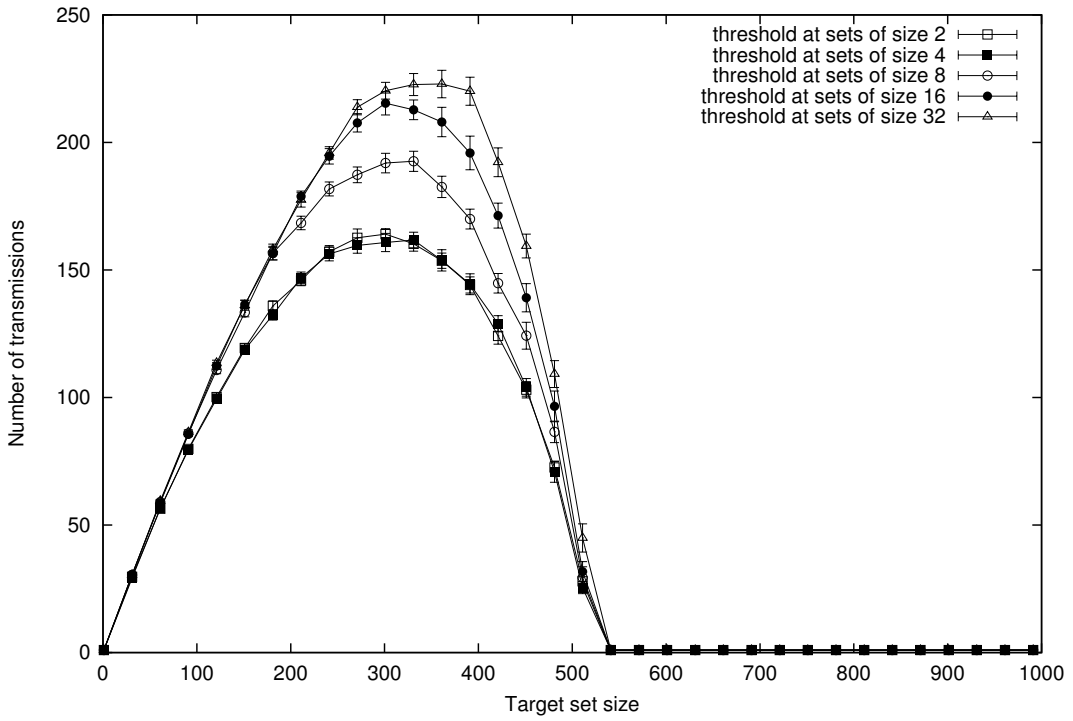

Figure IV.4: The effect of the "$\leq$" threshold T on the number of transmissions (t), for a tree with $n = 1024$.

we focus on simulations of small populations for another reason. We shall see in Section IV.F.4 that we can capitalize on the detailed understanding of small populations when we discuss partitioning large populations. Our results show that breaking a large population into small subgroups and solving the problem independently for each subgroup results in a good performance trade-off.

"$<$" or "$\leq$"?

A subtle issue in the execution of algorithm *f-Cover* is whether the inequality in step 2 is strict ($<$) or not ($\leq$). Assume that $f = 2$ and that the collection $\mathcal{S}$ is a full binary tree. If a set of size S_i with $|S_i| = 2$ is tested using non-strict inequality, and only one member of S_i is in the target set K, then S_i is selected as a candidate and may be part of the cover. However, using a strict inequality gives a better choice, which is to select the singleton containing that

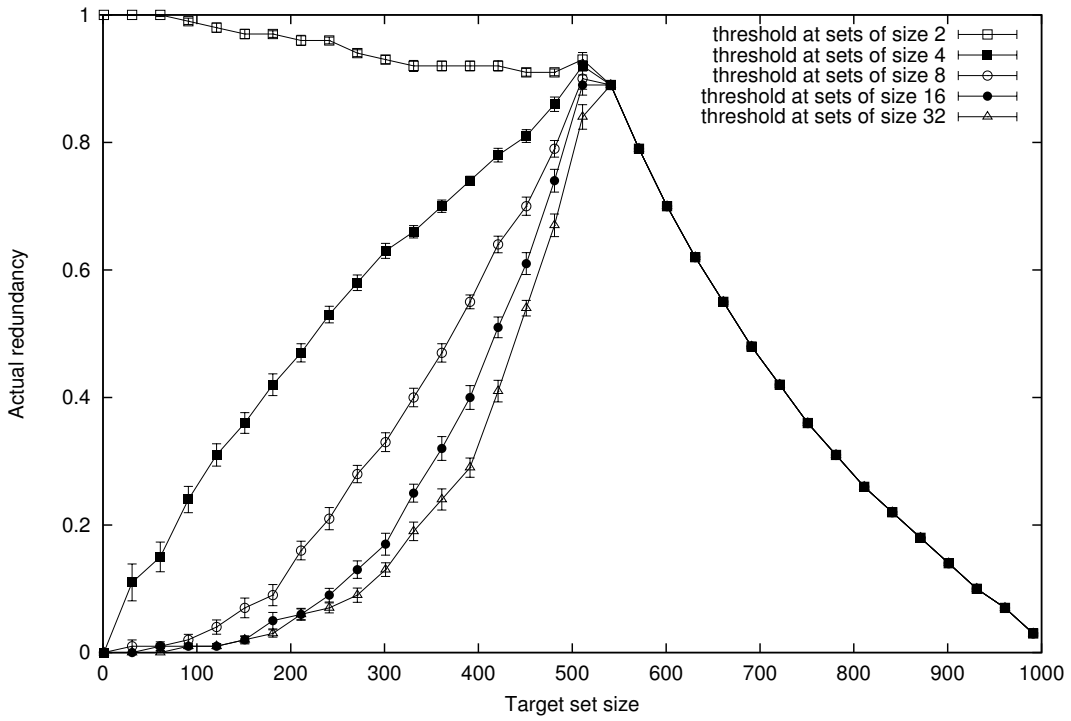

Figure IV.5: The effect of the "$\leq$" threshold T on the Actual redundancy (f_a) for a tree with $n = 1024$.

user, thereby reducing the actual redundancy without increasing the number of transmissions. On the other hand, using strict inequality for larger set sizes tends to increase the number of transmissions. So, intuitively, we would like to use "$<$" in the lowest levels of the tree, and use "$\leq$" for sets of size T or larger, for an appropriate threshold T. Figures IV.4, IV.5 and IV.6 compare the performance of a tree scheme when the threshold is varied. Note that the $T = 2$ curve, which we commented on before, represents using "$\leq$" everywhere.

The most striking graph is that of the actual redundancy (Figure IV.5). We see that when we use strict inequality in the level of the tree corresponding to sets of size 2 (i.e., the "$\leq$" threshold is $T = 4$) the actual redundancy, f_a, drops dramatically for target set sizes below $n/2$. At the same time, the number of transmissions, t, remains unchanged. There is also an improvement in the opportunity, η. Moving the threshold further up improves f_a and η at the cost of

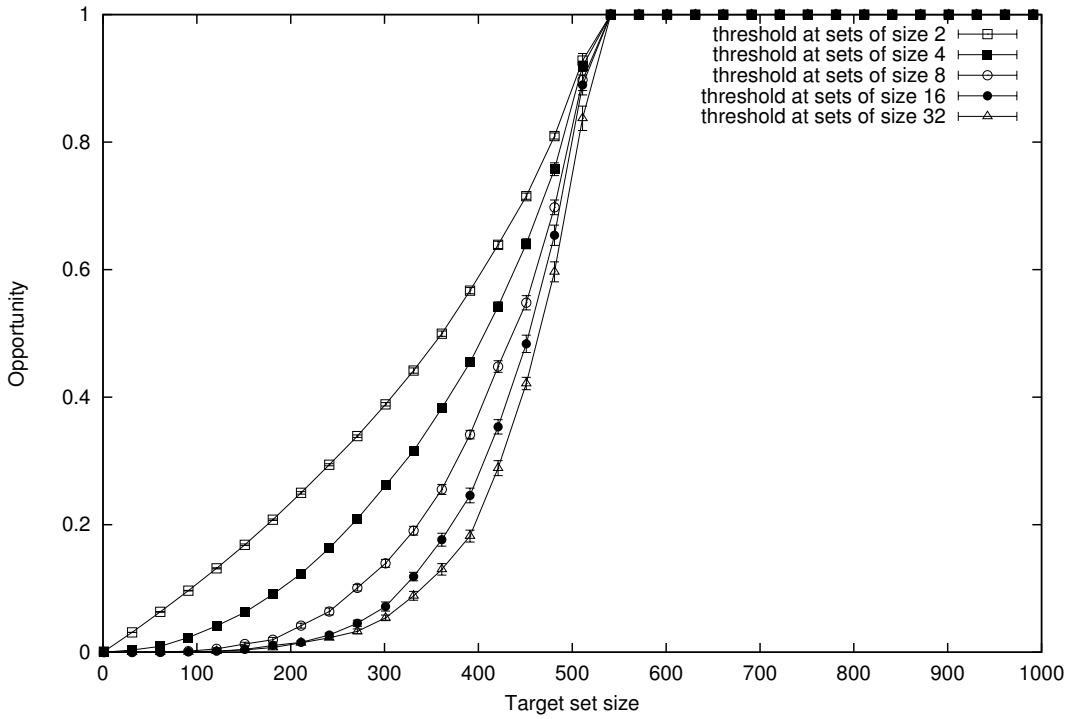

Figure IV.6: The effect of the "$\leq$" threshold T on the Opportunity (η), for a tree with $n = 1024$.

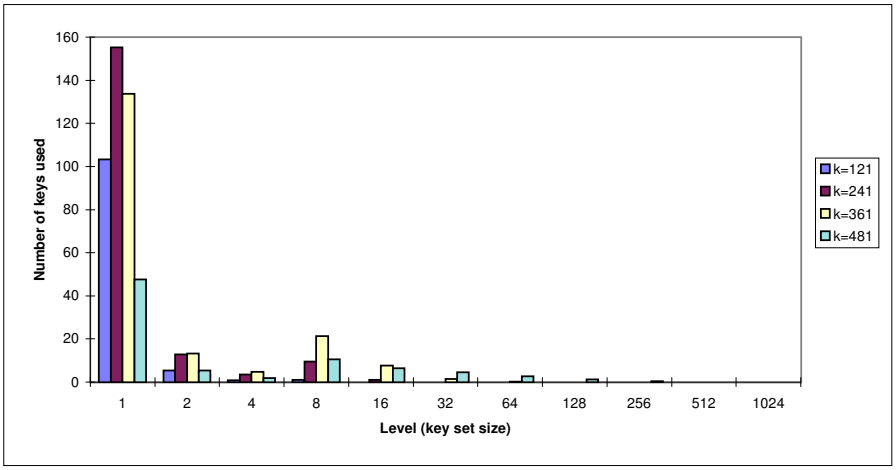

Figure IV.7: A histogram of the key sizes used for several target set sizes k, for $n = 1024$.

increasing t. We found out that, in most cases, and especially when extra keys are added (see below), it pays to set the threshold at $T = 8$ since the increase in t is very small while the gain in f_a and η is substantial. Thus, in all the following simulations we only use strict inequality for sets of size 4 and below.

Note that choosing $T = 8$ has an effect on the worst case performance since now $k = 3n/8$ users can be selected such that no four of them belong to a common set of size 8 and no three of them belong to a common set of size 4. As a result, we would be forced to use $t = 3n/8$ transmissions, all at the level corresponding to singleton sets.

When $T = 8$, the peak number of transmissions t is $193 \approx n/5$ (see Figure IV.4), which means a 50% improvement over the worst case performance of 384, and achieves actual redundancy that is always lower than 0.9. However, in most of the range the results are much better. In particular, if the interesting target set size range is below $k = n/5$, we get $t < n/6$, $f_a < 0.16$, and $\eta < 0.04$.

IV.F.3 Where Extra Keys are Effective

The basic tree scheme requires only $\log_2 n$ keys to be stored in each receiver. Therefore it is reasonable to consider schemes with slightly more keys: For

populations of several millions, we can afford to keep twice or four times as many keys in a receiver.

In this section, we study schemes in which a tree is augmented by additional sets. The motivation for doing so is clear: by increasing the number of sets (and thereby keys), the probability of finding a smaller cover increases. We are interested in locating the levels where it best pays to add sets, subject to the constraints on the number of keys per receiver.

In order to generate the extra key sets, we start with a "level-degree" profile, which specifies how many keys each user should hold at each level. For a level with set size k, a degree of d implies that each user should belong to $d-1$ extra sets, in addition to the one basic tree set it belongs to at this level. Thus we need to be able to generate nd/k sets of size k, such that each user belongs to exactly d of them. We achieve this by randomly permuting the n users $d-1$ times, and for each random permutation we add the users in positions $(i-1)k+1, \ldots, ik$ as a set, for $i = 1, \ldots, n/k$.

A vivid explanation for the preferred placement of the extra keys can be found in the histogram in Figure IV.7. The histogram depicts the number of keys used from each level of sets, for target sets of four sizes. We used a population of $n = 1024$ users and a basic tree scheme with 11 levels. The histogram clearly shows that the small sets are the ones used most often. As the target set size grows, some larger key sets are also used. However, even when the target sets are $k = 241$ and $k = 361$, i.e., target sets requiring the highest number of transmissions, relatively few keys are used for sets of size 32 and up. Therefore it seems that adding key sets at the low levels of the tree is the right approach.

Figures IV.8, IV.9, and IV.10 depict the performance of an 11-level tree ($n = 1024$) augmented tree with 9 extra keys. This choice allows us to double the number of keys per level in all the intermediate levels ($1 < |S_i| < n$). Following the conclusions we draw from the key usage histogram in Figure IV.7, these extra keys are distributed as uniformly as possible among the levels from the bottom

(couples) level up to some level ℓ. We varied ℓ in order to find the most effective distribution.

We first note that regardless of how the extra keys are distributed, the peak number of transmissions drops by at least 23% (from 193 down to 147 for the "up to sets of size 2" distribution) in comparison to a non-augmented tree.

Figure IV.8 shows that the best t is achieved by distributing the extra keys at the three lowest levels, i.e., adding couples, quadruplets, and octets. Adding sets of size 16 as well resulted in an almost identical performance. However, adding even larger sets gave significantly inferior performance. Figures IV.9 and IV.10 show that this improvement comes at the expense of an increase in f_a for small target set sizes, although the actual redundancy is still well below the guaranteed worst case of $f_a \leq f - 1$ ($= 1$ when $f = 2$).

In a similar experiment with 38 keys ($= 11 + 3 \times 9$) per user, the best t was achieved by spreading the keys among the lowest 4 levels (up to sets of size 16); the peak t for this experiment was about 94 transmissions, for target sets of size 271 ($= n/3.8$), which is 22% lower than the 121 achieved in Figure IV.8 by the "up to sets of size 8" distribution. We also ran the same experiments for larger and smaller values of n, with similar results. We omit the details.

Our conclusions from this set of experiments are that (a) adding a few extra keys per user substantially reduces the number of transmissions t, and (b) it pays to add these extra keys at the lower levels of the tree rather than to distribute them at higher levels as well.

IV.F.4 Partitioning

The results in the previous sections suggest that keys are more "valuable" at the lower levels of the tree than at the higher levels. Thus, it seems reasonable to discard the keys of the largest sets (highest levels) altogether, and to use the additional key space for more lower level keys. We achieve this by partitioning the population, n, into ν disjoint partitions of size n/ν. The space occupied by the

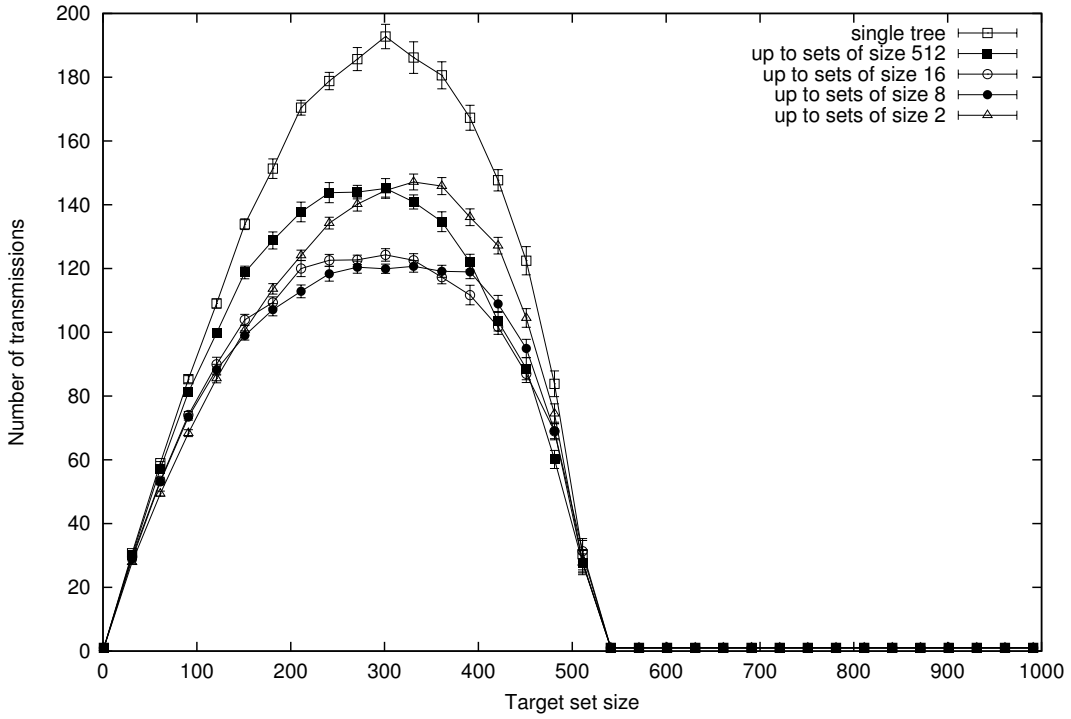

Figure IV.8: Number of transmissions (t) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

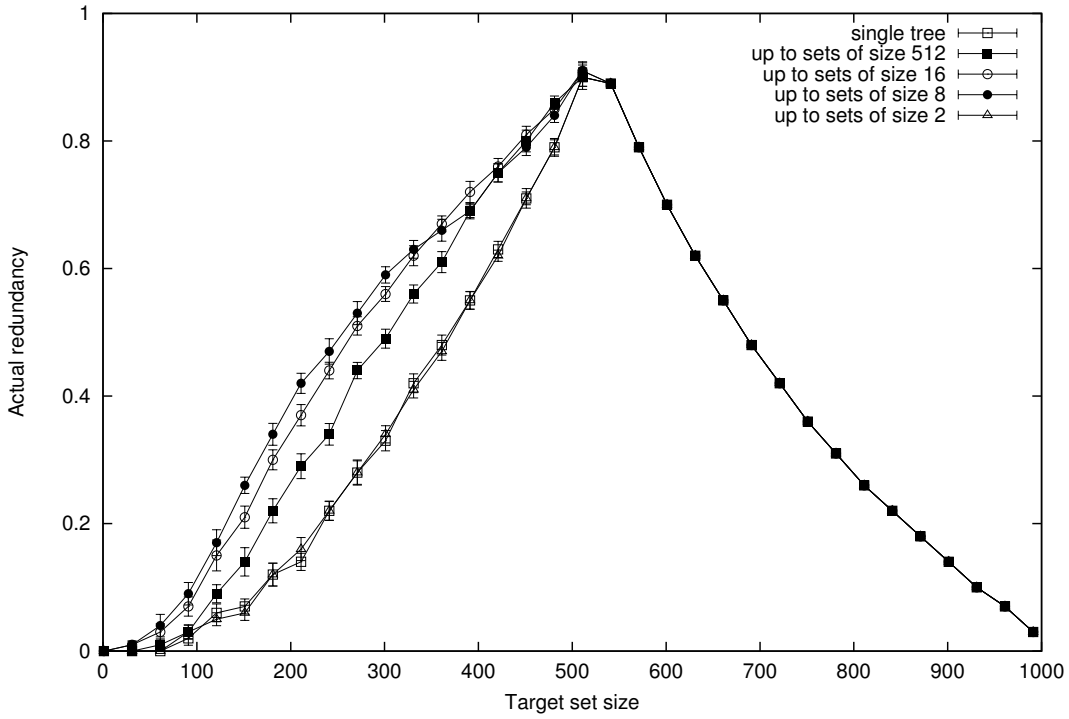

Figure IV.9: Actual redundancy (f_a) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

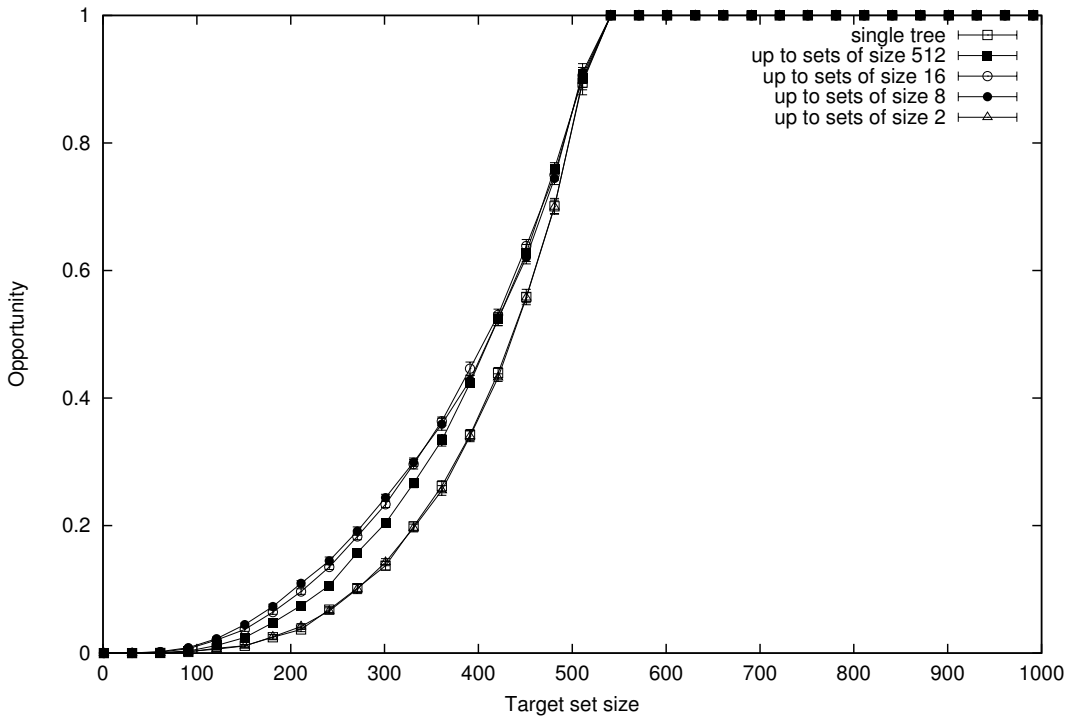

Figure IV.10: Opportunity (η) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

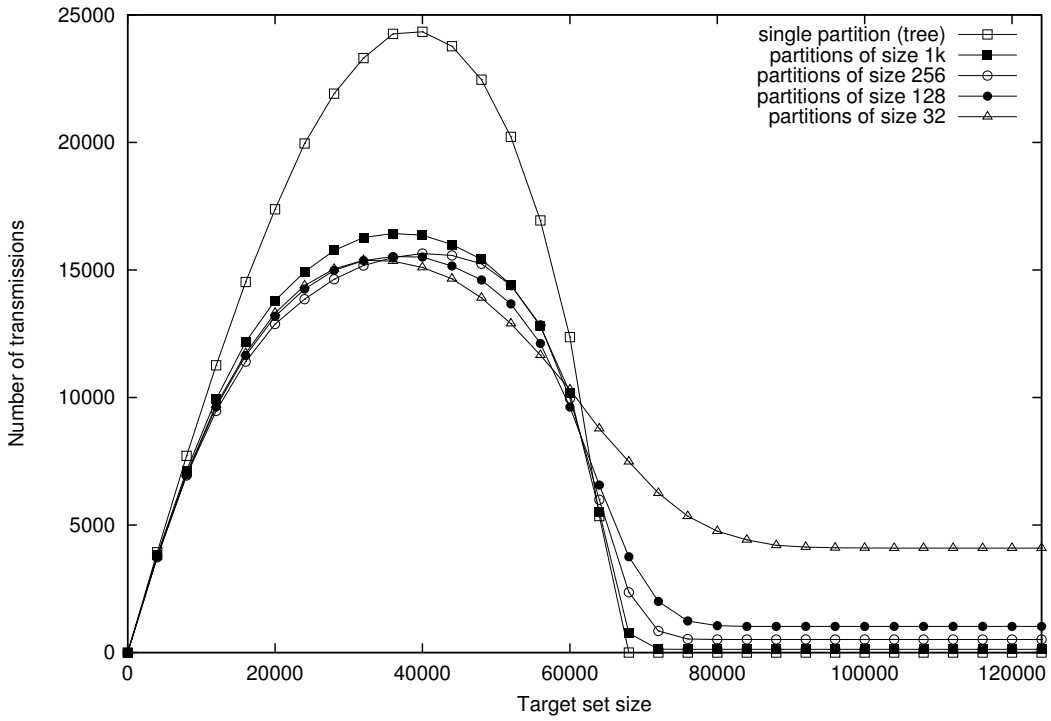

Figure IV.11: Number of transmissions (t) as a function of the target set size k, with $n = 128K$, $f = 2$, and 18 keys in total.

$\log_2 \nu$ deleted keys per user is then used to increase the number of low level sets in each partition.

In this section we concentrate on larger, more realistic user population sizes. However, since each individual partition is small, we can apply the insight we have gained from our earlier small-population experiments.

Figures IV.11 and IV.12 compare the performance of the a single tree scheme for a population of 128K customers with the performance of schemes that employ the same number of keys (18) but with ν partitions. Within each partition we distribute the $\log \nu$ extra keys to achieve the lowest peak t; as we have seen before, this means that the extra keys are distributed among the lowest levels in the tree, thus adding key sets of sizes between 2 and 32. For each value of ν we ran the equivalent of the experiment we discussed in Section IV.F.3. We report only the results of the best (lowest peak t) extra-key distribution for each value of

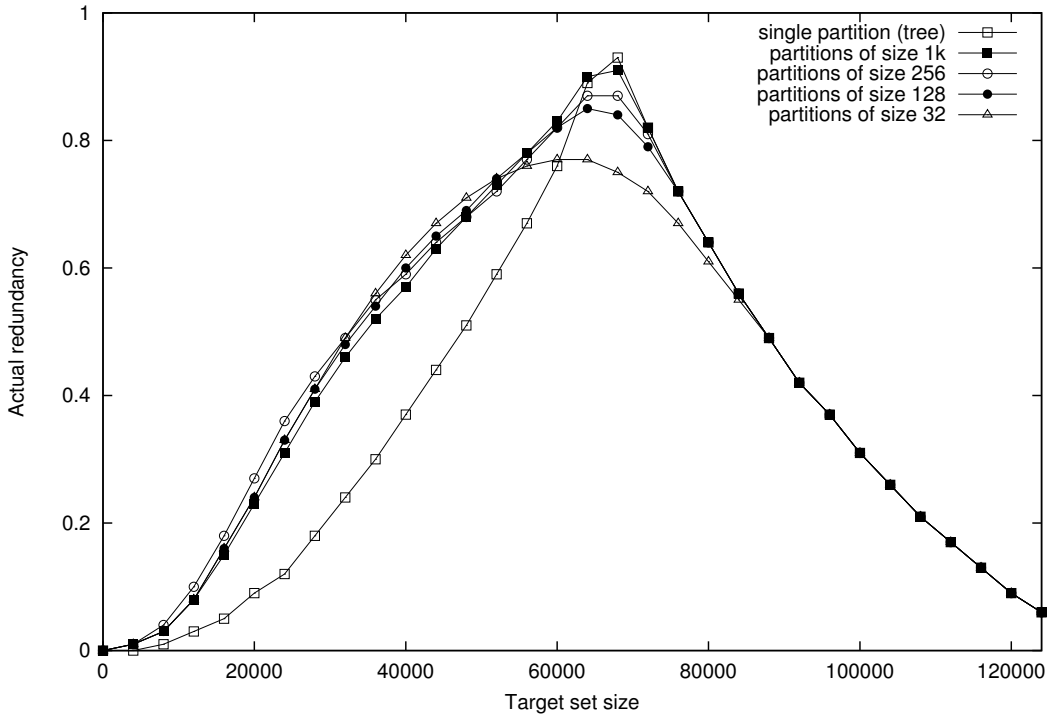

Figure IV.12: Actual redundancy (f_a) as a function of the target set size k, with $n = 128K$, $f = 2$, and 18 keys in total.

ν.

Figure IV.11 shows that the decrease in t is dramatic for a large range of target set sizes. In particular, the peak t drops by about 36%, from 24337 for a single partition to 15526 for $\nu = 1024$ partitions of size 128 each. Increasing the ν further reduces t for some values of k. However, for large target set sizes, and especially those with $k > n/2$, we pay a penalty in the number of transmissions. For such large target sets we have to use $t = \nu$ transmissions instead of one. We argue that as long as ν is substantially smaller than the peak t, the savings in t for smaller target sets far outweighs the penalty incurred for large target sets. Moreover, dealing with targets with $k > n/2$ can be done by maintaining a single additional broadcast key together with the partitions' keys.

Figure IV.12 shows that partitioning the users increases f_a for target sets with $k < n/2$. However, the peak f_a actually drops since we no longer use the

very large key sets, e.g., those with size $n/2$ or $n/4$. Partitioning also improves the opportunity for $k \sim n/2$ (graph omitted).

We conclude that partitioning the users is an effective method for designing establishment key allocations. It is better to discard the large high-level key sets in favor of extra sets at the low levels. As a rule of thumb we suggest to use at least $\nu \approx \sqrt{n}$ partitions, and possibly more for larger values of n.

IV.G Dynamic Environments

In this section we discuss aspects of implementing our key management scheme in a dynamic environment such as the Internet. We distinguish between two types of dynamics. One type is the population's dynamics, i.e., the change in the population as new users are added and dropped from the MBone infrastructure. The other type is in-program dynamics, i.e., the case where paying customers join and leave a specific program broadcast while it is taking place.

IV.G.1 Adapting to a Dynamic Population

Our first concern is how to adapt the static establishment key allocation we described before to a user population that is changing over time. We note that the user population is mainly growing as new networks are connected to the MBone. Networks depart from the MBone at a lower rate.

In Section IV.F.4 we showed that instead of building one monolithic establishment key allocation, it is better to split the population into partitions of a smaller size, say of 1024 users in each. Each of these partitions has its own establishment key allocation. Using this observation, we suggest building the establishment key allocation incrementally, in phases, as the population changes. A new partition is created at the beginning of each phase, with virtual "place holder" users. An establishment key allocation is constructed for the new partition, and each virtual user is assigned its keys. Each (real) new user that joins the MBone

replaces a virtual user, and is assigned the virtual user's keys. The phase ends when all the virtual users in the new partition have been replaced by real users. At this point a new phase starts.

A user disconnecting from the MBone (not a temporary logoff) is marked as non-existing. Once the number of non-existing users in a partition drops below some threshold, say half the users are non-existing, the partition is deleted and all the users are rekeyed to a new partition. Note that the cost of rekeying a user is amortized over all the leave operations that required no rekeying in the past.

The virtual users in a new partition and the non-existing users in old partitions are all accounted for when key covers are computed for specific programs. However, their presence can only make f_a better: some of the redundant users that are part the cover $\mathcal{C}(K)$ (for which the ratio f is guaranteed) are not really there.

IV.G.2 In-Program Dynamics

Next we discuss how decryption keys are transmitted in a dynamic environment, in which users join or leave a specific program, i.e., when the target set K is dynamically changing while the program is being transmitted. We believe that this type of fine-grained join/leave control is more appropriate in the MBone environment, where a single program may be quite lengthy.

To handle the dynamic changes in the target set, we divide the program's transmission time into slots of certain length, say 5 minutes. In each time slot, a different encryption key is used. We collect all the join/leave operations within slot $i-1$, compute the updated target set K_i, and recalculate the key cover $\mathcal{C}(K_i)$ for the next slot. The recipients in $R_{\mathcal{C}}(K_i)$ receive the new key.

Our goal is to quantify the effect of the in-program dynamics on the number of free-riders. To this end, we introduce the following definitions.

Definition IV.G.1 Consider the first i slots of a program transmission.

1. Let $\mathcal{V}^i$ denote the set of users that can view all the i slots.
2. Let $\mathcal{P}^i$ denote the set of users that pay for all the i slots.

3. Let $\mathcal{N}^i$ denote the set of users that do not pay for any of the i slots.

To measure the dynamic redundancy, we define the free-to-pay ratio $\rho(i)$, which is the proportion of non- and partial-paying users in the viewer set $\mathcal{V}^i$. Formally,

Definition IV.G.2 Let $\rho(i) = \frac{|\mathcal{V}^i \setminus \mathcal{P}^i|}{|\mathcal{V}^i|}$.

The ratio ρ is similar to a dynamic version of the actual redundancy f_a (recall Definition IV.B.6), but with a different denominator: $\rho(1) = (r_{\mathcal{C}}(K) - k)/r_{\mathcal{C}}(K)$ whereas $f_a = (r_{\mathcal{C}}(K) - k)/k$. For f-redundant establishment key allocations we have $\rho(1) \leq 1 - 1/f$, however, this inequality may not hold for larger values of i since $\mathcal{V}^i$ and $\mathcal{P}^i$ evolve at different rates.

We define the dynamic opportunity, η_d, as the proportion of non-paying recipients (free-riders) in the non-paying population.

Definition IV.G.3 Let $\eta_d(i) = \frac{|\mathcal{V}^i \cap \mathcal{N}^i|}{|\mathcal{N}^i|}$.

This is a generalization of the opportunity η of Definition IV.B.7, and $\eta_d(1) = \eta$.

IV.G.3 Experimenting with In-Program Dynamics

We conducted a series of simulation experiments to evaluate the effect of recalculating the key cover for every slot on the possibility to receive a program for free. We focused on relatively small user populations, since, as we have seen in Section IV.F.4, partitioning the population into many small partitions is advantageous.

The results we report here are all for $n = 1024$. We used 20 keys per user, using the best scheme we found in the experiments of Section IV.F.3, namely, distributing the 9 extra keys up to sets of size 8. We experimented with other values of n, and with other key distributions, with essentially the same results, so we omit the details.

The in-program dynamics are captured by the following two parameters:

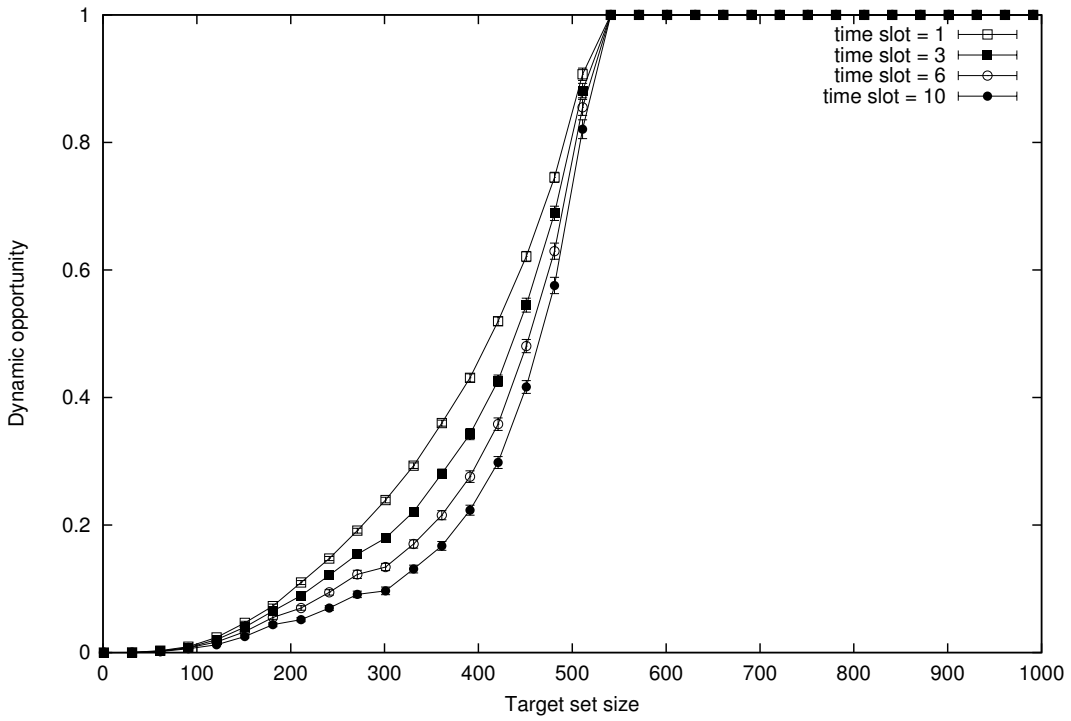

Figure IV.13: The dynamic opportunity $\eta_d(i)$ for $i = 1, 3, 6, 10$ slots, as a function of the target set size k, using $\tau = 0.01$ for $n = 1024$.

- $|K_1|$ is the initial paying population size;
- τ is the fraction of the paying population at slot i that stops paying during slot i (leaving users);

For simplicity we assume that every user that leaves in slot i is replaced by a joining user. Thus $|K_i| = |K_1|$ for all i. Exactly $\tau|K_{i-1}|$ leaving users are chosen at random from the set K_{i-1}, and the same number of joining users are chosen at random from $\mathcal{U} \setminus K_{i-1}$. The values we tested for τ were 0.01, 0.02, 0.05, and 0.1.

Figures IV.13 and IV.14 show the effect of the in-program dynamics on the dynamic opportunity $\eta_d(i)$. We see that even when the target set size changes by only 1% in each slot, the dynamic opportunity drops by between 33% and 52% for $250 \leq k \leq 450$ by the end of ten slots. When the target set size changes by 10% in each slot (Figure IV.14), the dynamic opportunity becomes negligible (less

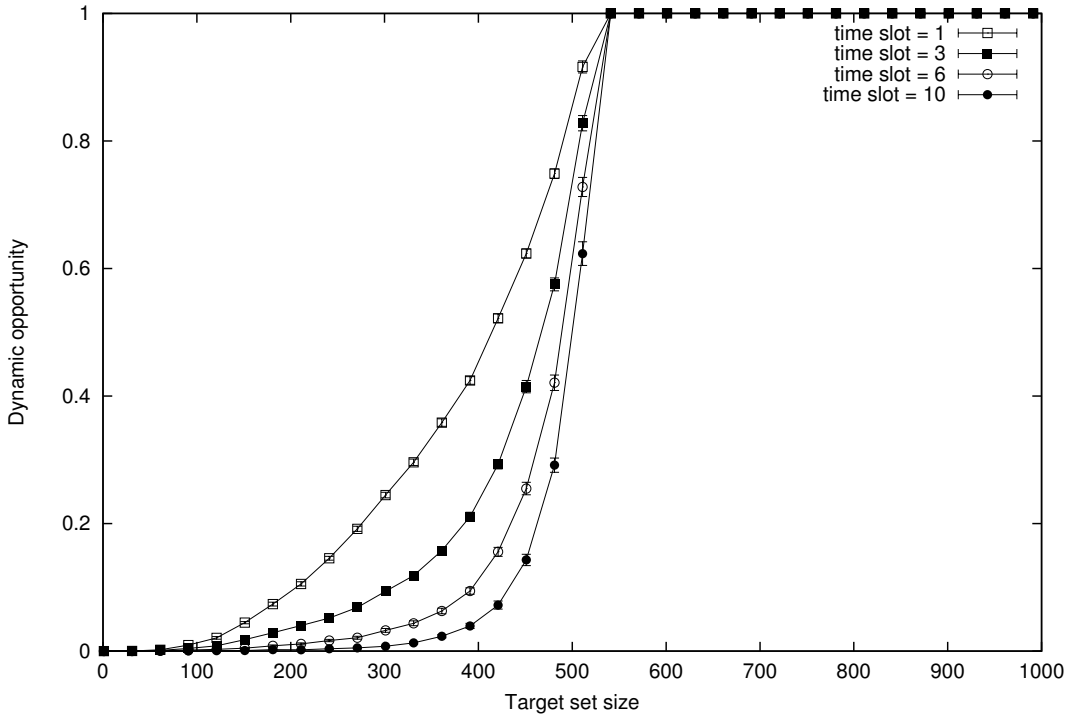

Figure IV.14: The dynamic opportunity $\eta_d(i)$ for $i = 1, 3, 6, 10$ slots, as a function of the target set size k, using $\tau = 0.1$ for $n = 1024$.

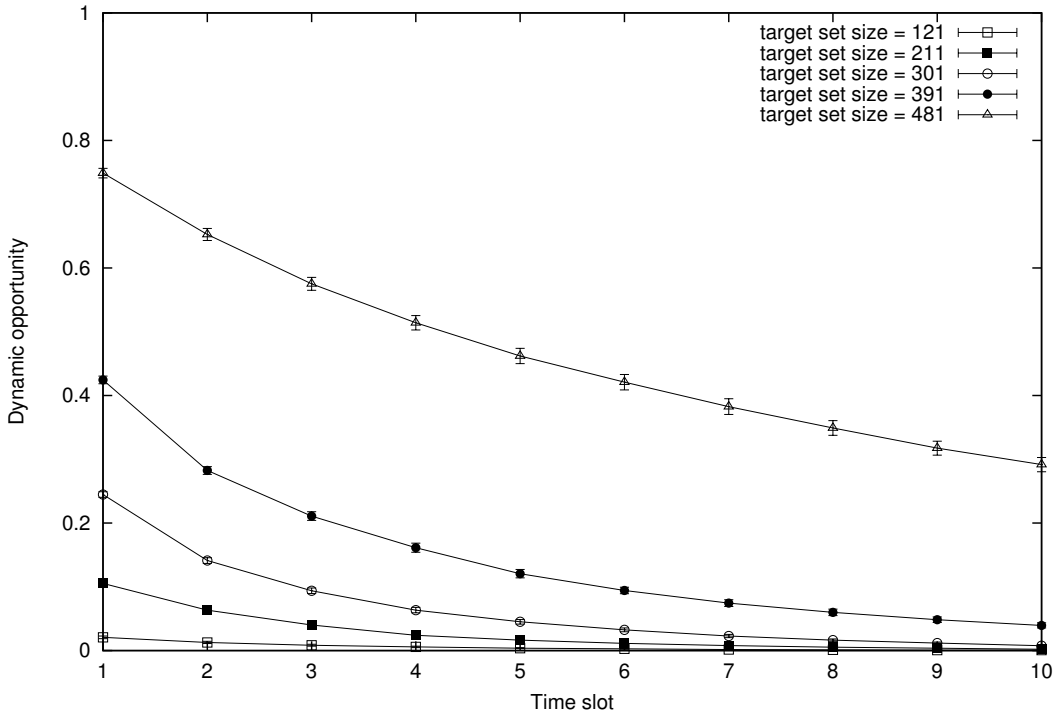

Figure IV.15: The dynamic opportunity $\eta_d(i)$ for several target set sizes, as a function of the slot number i, using $\tau = 0.1$ for $n = 1024$.

than 2%) after ten slots, for all target set sizes below 30% of the population. It is apparent from the figures that $\eta_d(i)$, as a function of k, tends to a step function when $i \to \infty$. The rate of the convergence to a step function depends on τ, with higher values of τ resulting in faster convergence.

The explanation for this big gain, even for small change rates, lies in the algorithm we use to find the key cover (recall Figure IV.3). In step 3 of the algorithm, we pick a set that minimizes the actual redundancy. In many cases there are several choices for this set, and the arbitrary tie-breaking selection made by the algorithm determines the set of free-riders. A small random perturbation in the target set K randomizes the tie-breaking, resulting in a significant change in the set of free-riders.

Figure IV.15 shows the rates by which $\eta_d(i)$ drops as a function of time for $\tau = 10\%$. The dynamic opportunity drops to negligible levels for all but the

largest target sets by the end of only 10 time slots. When the rate of change is 1% (graph omitted) the decrease is slower, however, as we said in the discussion of Figure IV.13, the drop is still substantial in the mid-sized target sets.

Figure IV.16 shows the dramatic drop in the dynamic opportunity when the rate of change τ grows from 1% to 10%, in comparison to the opportunity η for a static target set. We see that even when the dynamics are minimal ($\tau = 0.01$) there is a 60% drop in the opportunity, e.g., from $\eta = 0.24$ to $\eta_d(10) = 0.097$ for $k = 301$. For higher rates of change the drop is even more pronounced.

We conclude from all the above discussion that the in-program dynamics makes the service provider's situation more favorable. The changing target population results in significantly better performance from our f-redundant establishment key allocations: the free-rider's opportunity rapidly decreases, even for low change rates. Thus it becomes increasingly hard to be able to watch an entire program for free, as the number of slots increases.

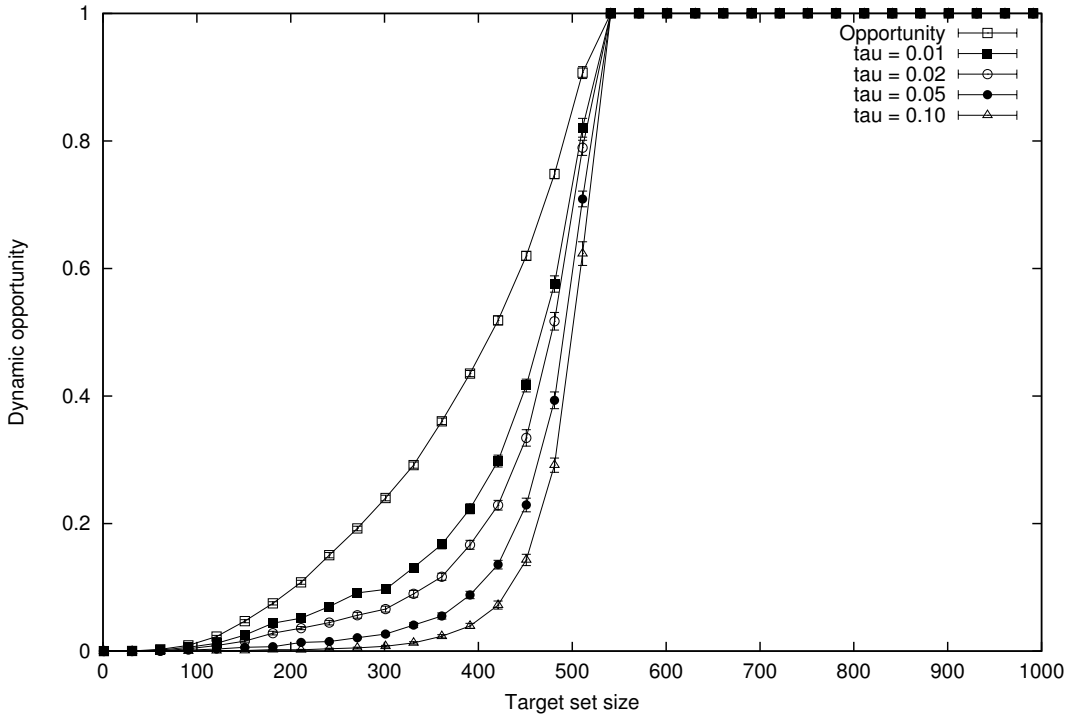

Figure IV.16: The dynamic opportunity $\eta_d(10)$, after 10 slots, for different values of τ, as a function of the target set size k, for $n = 1024$.

Bibliography

[1] M. Abdalla and M. Bellare. Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 546–559. Springer-Verlag, Berlin Germany, December 3–7, 2000.

[2] M. Abdalla, M. Bellare, and P. Rogaway. DHIES: an encryption scheme based on the Diffie-Hellman problem. Contributions to IEEE P1363a, September 1998.

[3] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158, April 8–12,, San Francisco, USA 2001. Springer-Verlag, Berlin Germany.

[4] M. Abdalla, Y. Shavitt, and A. Wool. Towards making broadcast encryption practical. In M. Franklin, editor, *Financial Cryptography'99*, volume 1648 of *Lecture Notes in Computer Science*, pages 140–157, Anguilla, BWI, February 1999. Springer-Verlag, Berlin Germany.

[5] M. Abdalla, Y. Shavitt, and A. Wool. Key management for restricted multicast using broadcast encryption. *IEEE/ACM Transactions on Networking*, 8(4):443–454, August 2000. `http://www.michelabdalla.net/pubs`.

[6] American National Standards Institute (ANSI) X9.F1 subcommittee. ANSI X9.63 Public key cryptography for the Financial Services Industry: Elliptic curve key agreement and key transport schemes, July 5, 1998. Working draft version 2.0.

[7] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1996. Springer-Verlag, Berlin Germany.

[8] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In IEEE, editor, *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, 1997.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[10] M. Bellare, O. Goldreich, and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In M. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1999. Springer-Verlag, Berlin Germany.

[11] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 21–25, 1994. Springer-Verlag, Berlin Germany.

[12] M. Bellare, J. Kilian, and P. Rogaway. Luby-rackoff backwards: Increasing security by making block ciphers non-invertible. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1996.

[13] M. Bellare and S. Miner. A forward-secure digital signature scheme. In M. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Santa Barbara, CA, USA, August 1999. Springer-Verlag, Berlin Germany.

[14] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, Berlin Germany, December 3–7, 2000.

[15] M. Bellare and P Rogaway. Optimal asymmetric encryption: How to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1994. `http://www-cse.ucsd.edu/users/mihir`.

[16] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and rabin. In U. Maurer, editor, *Advances in Cryptol-*

ogy – EUROCRYPT'96, volume 1070 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1996.

[17] M. Bellare and P. Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In *Information and Communications Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin Germany, 1997.

[18] M. Bellare and B. Yee. Forward security in private key cryptography. Cryptology ePrint Archive, Report 2001/035, 2001. `http://eprint.iacr.org/`.

[19] E. Biham and A. Shamir. Differential cryptanalysis of the full 16-round des. In E. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1992. Springer-Verlag, Berlin Germany.

[20] D. Bleichenbacher. A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[21] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[22] C. Blundo and A. Cresti. Space requirements for broadcast encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 287–298. Springer-Verlag, Berlin Germany, May 1994.

[23] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Generalized Beimel-Chor schemes for broadcast encryption and interactive key distribution. *Theoretical Computer Science*, 200(1–2):313–334, 1998.

[24] D. Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, 1998. Invited paper.

[25] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1996. Springer-Verlag, Berlin Germany.

[26] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *30th Annual ACM Symposium on Theory of Computing*, New York, NY, May 23–26, 1998. ACM Press.

[27] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270, Santa Barbara, CA, USA, August 21–25, 1994. Springer-Verlag, Berlin Germany.

[28] J. Cohen, M. Etzel, D. Faucher, and D. Heer. Security for broadband digital networks. *Communications Technology*, pages 58–69, August 1995.

[29] C. Colborn and J. Dinitz, editors. *The CRC Handbook of Combinatorial Designs*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, fifth edition, 1996.

[30] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[31] C. Dwork D. Dolev and M. Naor. Non-malleable cryptography. In ACM, editor, *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, May 6–8, 1991. ACM Press.

[32] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1978.

[33] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, June 1992.

[34] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography, 1998. Manuscript.

[35] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[36] H. Eriksson. MBone: the multicast backbone. *Communications of the Association for Computing Machinery*, 37(8):54–60, August 1994.

[37] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the Association for Computing Machinery*, 45(4):634–652, 1998.

[38] A. Fiat and M. Naor. Broadcast encryption. In D. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491, Santa Barbara, CA, USA, August 1994. Springer-Verlag, Berlin Germany.

[39] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology*

– *CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1999. Springer-Verlag, Berlin Germany.

[40] E. Gabber and A. Wool. On location-restricted services. *IEEE Networks Magazine*, 13(6):44–52, November/December 1999.

[41] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[42] O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *IACR Journal of Cryptology*, 6(1):21–53, 1993.

[43] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33:792–807, 1986.

[44] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[45] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Euro. J. Operational Research*, 101(1):81–92, August 1997.

[46] C. Günther. An identity-based key-exchange protocol. In J-J. Quisquater and J. Vandewille, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer-Verlag, Berlin Germany, May 1989.

[47] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[48] C. Hall, D. Wagner, J. Kelsey, and B. Schneier. Building PRFs from PRPs. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 370–389, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[49] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, MA, 1995.

[50] IEEE P1363a Committee. IEEE P1363a / D9 — standard specifications for public key cryptography: Additional techniques. `http://grouper.ieee.org/groups/1363/index.html/`, June 2001. Draft Version 9.

[51] D. Johnson and M. Peyravian S. Matyas. Encryption of long blocks using a short-block encryption procedure, November 1996. `http://stdsbbs.ieee.org/groups/1363/index.html`.

[52] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Computer System Sci.*, 9:256–278, 1974.

[53] D. Kravitz and D. Goldschlag. Conditional access concepts and principles. In M. Franklin, editor, *Financial Cryptography'99*, volume 1648 of *Lecture Notes in Computer Science*, Anguilla, BWI, February 1999. Springer-Verlag, Berlin Germany.

[54] A. M. Law and W. D. Kelton. *Simulation Modelling and Analysis*. McGraw-Hill, 2nd edition, 1991.

[55] C. Lim and P. Lee. Another method for attaining security against adaptively chosen ciphertext attacks. In D. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1994. Springer-Verlag, Berlin Germany.

[56] L. Lovász. On the ratio of optimal integral and fractional covers. *Disc. Math.*, 13:383–390, 1975.

[57] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 512–526. Springer-Verlag, Berlin Germany, May 1998.

[58] B. M. Macq and J.-J. Quisquater. Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957, 1995.

[59] M. Matsui. The first experimental cryptanalysis of the data encryption standard. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 21–25, 1994. Springer-Verlag, Berlin Germany.

[60] J. McCormac. *European Scrambling Systems 5*. Waterford University Press, Waterford, Ireland, 1996.

[61] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988. Special issue on cryptography.

[62] W. H. Mills and R. C. Mullin. Coverings and packings. In J. H. Dinitz and D. R. Stinson, editors, *Contemporary Design Theory: A Collection of Surveys*, pages 317–399. John Wiley & Sons, 1992.

[63] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM*, September 1997.

[64] M. Naor and B. Pinkas. Threshold traitor tracing. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in*

Computer Science, Santa Barbara, CA, USA, August 1998. Springer-Verlag, Berlin Germany.

[65] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In IEEE, editor, *38th Annual Symposium on Foundations of Computer Science*, Miami Beach, FL, October 19 - 22, 1997. IEEE, IEEE Computer Society Press.

[66] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In ACM, editor, *22nd Annual ACM Symposium on Theory of Computing*, Baltimore, Maryland, May 14–16, 1990. ACM Press.

[67] PKCS #1: RSA cryptography standard. RSA Data Security, Inc., June 1991.

[68] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 1991. Springer-Verlag, Berlin Germany.

[69] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[70] J. Schönheim. On coverings. *Pacific J. Math.*, 14:1405–1411, 1964.

[71] Certicom research, standards for efficient cryptography group (SECG) — sec 1: Elliptic curve cryptography. `http://www.secg.org/secg_docs.htm`, September 20, 2000. Version 1.0.

[72] Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, April 1995.

[73] C. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[74] V. Shoup. Personal Communication.

[75] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1997.

[76] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 2000.

[77] D. R. Stinson and T. van Trung. Some new results on key distribution patterns and broadcast encryption. *Designs, Codes and Cryptography*, 14(3):261–279, 1998.

[78] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. Internet Draft, September 1998. Available from `http://www.ietf.org/ID.html`.

[79] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[80] Avishai Wool. Key management for encrypted broadcast. *ACM Transactions on Information and System Security*, 3(2), May 2000.

[81] A. Yao. Theory and applications of trapdoor functions. In IEEE, editor, *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, 1982.

[82] Y. Zheng. Public key authenticated encryption schemes using universal hashing. Contribution to P1363. `ftp://stdsbbs.ieee.org/pub/p1363/contributions/aes-uhf.ps`.

[83] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attack. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.