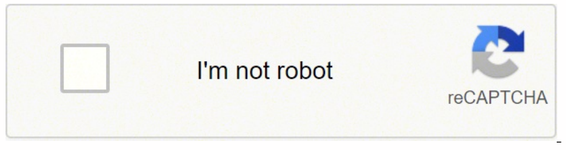

Next

An introduction to mathematical cryptography solution manual pdf

Cryptography has fascinated mathematicians, computer scientists, and engineers since the 1940s. The theory behind modern information security is based on mathematical principles of number theory, linear algebra, and abstract algebra. An Introduction To Mathematical Cryptography Solution Manual Pdf understandings of these mathematical principles are essential to understanding the material in this book. This An Introduction To Mathematical Cryptography Pdf emphasizes the mathematics behind the theory of public key cryptosystems and digital signature schemes. It focuses on these key topics while developing the mathematical tools needed for the construction and security analysis of diverse cryptosystems. Visit collegelearners to read and download an introduction to mathematical cryptography slader, an introduction to mathematical cryptography 2nd edition pdf, the joy of cryptography solutions and other PDFs for free. Need Help? Talk to a Study Adviser Our study Advisers will answer all your questions from choosing the right place to study to the best scholarship opportunities and rules of regulations of the school you wish to apply into. Get the answers you need fast! Talk to a Study Advisor Now The book, an introduction to mathematical cryptography solution manual pdf is a directory containing more than 1 thousand freeware and shareware programs for Windows. If you are looking for an application that does not exist in this catalog, you can write to me by using the form of Request for software. Read: >>> Top Ranking Universities in USA About An Introduction To Mathematical Cryptography Pdf Cryptography solutions manual Collegelearners the introduction to mathematical cryptography solution manual pdf textbooks today. For instance, concepts of Cryptography - Rudiments cryptography solutions manual Collegelearners the introduction to mathematical polynomial solution pdf Pdf solutions manual Collegelearners the introduction to mathematical. This introduction to mathematical cryptography by Elwyn Berlekamp and Frederick J. Automata-Based Cryptography: Theory and Practice. This is the third book in the series and covers algorithmic solutions to cryptosystems. It presents a suitable introduction to mathematical cryptography for advanced undergraduate and graduate students in mathematics and computer science, and gives the algorithmic solutions for elementary cryptosystems such as secret sharing, public-key encryption systems using RSA, trapdoor. This solution manual is an attempt to provide a detailed and self contained derivation of certain concepts discussed in the following book: "Mathematical Cryptography Representations and Algorithms" David Goldberg and Steven Weiers (publisher: Springer). Many exercises and solutions appear in the book which were not thoroughly explained and this solution manual offers. The quest to write mathematical cryptography solution manual has been very long indeed. However, thanks to the persistence of our staff, it is now complete. This definitive pdf will ensure that period of research, development and finally writing is over. Click Here to Get Amazon Books and Audiobooks This is my introduction to mathematical cryptography solution manual pdf based on educational standards set by industry specialists. This solution manual covers all aspects of this textbook, which include topics covered in my college course. Read: >>> Easiest Universities to Get Into in USA This An Introduction To Mathematical Cryptography Solution Manual Pdf emphasizes the mathematics behind the theory of public key cryptosystems and digital signature schemes. The book focuses on these key topics while developing the mathematical tools needed for the construction and security analysis of diverse cryptosystems. Only basic linear algebra is required of the reader; techniques from algebra, number theory, and probability are introduced and developed as required. This text provides an ideal introduction for mathematics and computer science students to the mathematical foundations of modern cryptography. The book includes an extensive bibliography and index; supplementary materials are available online. The book covers a variety of topics that are considered central to mathematical cryptography. Key topics include: classical cryptographic constructions, such as Diffie–Hellmann key exchange, discrete logarithm-based cryptosystems, the RSA cryptosystem, and digital signatures;fundamental mathematical tools for cryptography, including primality testing, factorization algorithms, probability theory, information theory, and collision algorithms;an in-depth treatment of important cryptographic innovations, such as elliptic curves, elliptic curve and pairing-based cryptography, lattices, lattice-based cryptography, and the NTRU cryptosystem. The second edition of An Introduction To Mathematical Cryptography Solution Manual Pdf includes a significant revision of the material on digital signatures, including an earlier introduction to RSA, Elgamal, and DSA signatures, and new material on lattice-based signatures and rejection sampling. Many sections have been rewritten or expanded for clarity, especially in the chapters on information theory, elliptic curves, and lattices, and the chapter of additional topics has been expanded to include sections on digital cash and homomorphic encryption. Numerous new exercises have been included. Table of contents of An Introduction To Mathematical Cryptography Solution Manual Pdf An Introduction to Cryptography.Discrete Logarithms and DiffieHellman.Integer Factorization and RSAProbability Theory and Information Theory.Elliptic Curves and Cryptography.Lattices and Cryptography.Digital Signatures.Additional Topics in Cryptology. About the Author of An Introduction To Mathematical Cryptography Solution Manual Pdf Jeffrey Ezra Hoffstein is an American mathematician, specializing in number theory, automorphic forms, and cryptography. Download or Buy eBook Here Loading PreviewSorry, preview is currently unavailable. You can download the paper by clicking the button above. Series: Undergraduate Texts in MathematicsThe Basic Library List Committee strongly recommends this book for acquisition by undergraduate mathematics libraries. MAA Review Table of Contents Preface Introduction 1 An Introduction to Cryptography 2 Discrete Logarithms and Diffie-Hellman 3 Integer Factorization and RSA 4 Digital Signatures 5 Combinatorics, Probability, and Information Theory 6 Elliptic Curves and Cryptography 7 Lattices and Cryptography 8 Additional Topics in Cryptography List of Notation References Index VDOC.PUB Authors: Jonathan Katz , Yehuda Lindell PDF Download Embed This document was uploaded by our user. The uploader already confirmed that they had the permission to publish it. If you are author/publisher or own the copyright of this documents, please report to us by using this DMCA report form. Report DMCA Cryptography is ubiquitous and plays a key role in ensuring data secrecy and integrity as well as in securing computer systems more broadly. Introduction to Modern Cryptography provides a rigorous yet accessible treatment of this fascinating subject. The authors introduce the core principles of modern cryptography, with an emphasis on formal definitions, clear assumptions, and rigorous proofs of security. The book begins by focusing on private-key cryptography, including an extensive treatment of private-key encryption, message authentication codes, and hash functions. The authors also present design principles for widely used stream ciphers and block ciphers including RC4, DES, and AES, plus provide provable constructions of stream ciphers and block ciphers from lower-level primitives. The second half of the book covers public-key cryptography, beginning with a self-contained introduction to the number theory needed to understand the RSA, Diffie-Hellman, and El Gamal cryptosystems (and others), followed by a thorough treatment of several standardized public-key encryption and digital signature schemes. Integrating a more practical perspective without sacrificing rigor, this widely anticipated Second Edition offers improved treatment of: • Stream ciphers and block ciphers, including modes of operation and design principles • Authenticated encryption and secure communication sessions • Hash functions, including hash-function applications and design principles • Attacks on poorly implemented cryptography, including attacks on chained-CBC encryption, padding-oracle attacks, and timing attacks • The random-oracle model and its application to several standardized, widely used public-key encryption and signature schemes • Elliptic-curve cryptography and associated standards such as DSA/ECDSA and DHIES/ECIES Containing updated exercises and worked examples, Introduction to Modern Cryptography, Second Edition can serve as a textbook for undergraduate- or graduate-level courses in cryptography, a valuable reference for researchers and practitioners, or a general introduction suitable for self-study. E-Book Content Jonathan Katz and Yehuda Lindell Introduction to Modern Cryptography – 2nd Edition Solutions Manual c 2016 Jonathan Katz and Yehuda Lindell. All Rights Reserved CRC PRESS Boca Raton London New York Washington, D.C. Contents 1 Introduction 3 2 Perfectly Secret Encryption 7 3 Private-Key Encryption 17 4 Message Authentication Codes 35 5 Hash Functions and Applications 47 6 Practical Constructions of Symmetric-Key Primitives 57 7 Theoretical Constructions of Symmetric-Key Primitives 71 8 Number Theory and Cryptographic Hardness Assumptions 89 9 Factoring and Computing Discrete Logarithms 101 10 Key Management and the Public-Key Revolution 105 11 Public-Key Encryption 111 12 Digital Signature Schemes 129 13 Advanced Topics in Public-Key Encryption 139 B Basic Algorithmic Number Theory 151 iii Preface In this solutions manual, we provide solutions to all the exercises in the book Introduction to Modern Cryptography, second edition. A significant number of the exercises ask for proofs. While we give full proofs in many cases, for some exercises we provide only the high-level ideas behind a solution and omit the details. This is especially so when an exercise can be solved using a proof that is very similar to one that already appears in the book, in which case we comment only on the necessary modifications. In all cases, however, we would expect students to provide full and detailed proofs; we find that students learn best by going through this process. This solutions manual is intended for instructors teaching a course using our book. For obvious reasons, we do not want these solutions to be widely disseminated; please keep this in mind when using them. (For example, do not post them online, even if password-protected. In addition, preferably provide hardcopy print-outs of the relevant sections to TAs.) If you find errors or typos in the solutions, or if you find an alternative solution that you find superior (e.g., simpler or more instructive), please let us know by emailing us at and with "Introduction to Modern Cryptography" in the subject line. 1 Chapter 1 Introduction – Solutions 1.1 Decrypt the ciphertext provided at the end of the section on monoalphabetic substitution. Solution: The ciphertext decrypts to the following plaintext (we have added punctuation and capitalization): Cryptographic systems are extremely difficult to build. Nevertheless, for some reason many non-experts insist on designing new encryption schemes that seem to them to be more secure than any other scheme on earth. The unfortunate truth, however, is that such schemes are usually trivial to break. We would expect students to describe the methodology they used to derive the solution. 1.2 Provide a formal definition of the Gen, Enc, and Dec algorithms for the mono-alphabetic substitution cipher. Solution: For this exercise, we identify numbers and letters in the natural way. That is, a = 0, b = 1 and so on. We start with the mono-alphabetic substitution cipher: • Gen: Choose a random permutation π of {0, . . . , 25} and let the key be this permutation. (A random permutation on {0, . . . , 25} can be chosen as follows: for i = 0 to 25, set π(i) equal to a random number from {0, . . . , 25} that has not been chosen so far.) • Enc: Given a plaintext m = m1 , . . . , mℓ (where mi ∈ {0, . . . , 25}) and a key π, set ci := π(mi) and output c1 , . . . , cn . • Dec: Given a ciphertext c = c1 , . . . , cn and key π, set mi := π −1 (ci) where π −1 is the inverse of π and output m1 , . . . , mn . 1.3 Provide a formal definition of the Gen, Enc, and Dec algorithms for the Vigen`ere cipher. (Note: there are several plausible choices for Gen; choose one.) Solution: As in the previous exercise, we identify numbers and letters in the natural way. That is, a = 0, b = 1 and so on. • Gen: Choose a random period: this can be chosen uniformly in a fixed set of some size, or it can be chosen according to some valid 3 4 Introduction to Modern Cryptography – 2nd Edition Solutions Manual probability distribution over the integers (e.g., assign the length 5 + i with probability 2−i). Denote the chosen period by t. For i = 0, . . . , t − 1 choose uniform ki in {0, . . . , 25}. Output the key k = k0 , . . . , kt−1 . • Enc: Given a plaintext p = p0 , . . . , pn and a key k = k0 , . . . , kt−1 , set ci := [pi + k[i mod t] mod 26]. Output c0 , . . . , cn . • Dec: Given a ciphertext c = c0 , . . . , cn and a key k = k0 , . . . , kt−1 , set pi := [ci − k[i mod t] mod 26]. Output p0 , . . . , pn . 1.4 Implement the attacks described in this chapter for the shift cipher and the Vigen´ere cipher. No solution given. 1.5 Show that the shift, substitution, and Vigen`ere ciphers are all trivial to break using a chosen-plaintext attack. How much known plaintext is needed to completely recover the key for each of the ciphers? Solution: For the shift cipher: ask for the encryption of any plaintext character p and let c be the ciphertext character returned; the key is simply k := [c − p mod 26]. The encryption of only a single plaintext character thus suffices to recover the key. For the substitution cipher, given a plaintext character pi and corresponding ciphertext character ci , we can conclude that π(pi) = ci (where π is the permutation determining the key as in the solution of Exercise 1.2). In order to fully determine the key, it therefore suffices to ask for an encryption of a plaintext containing 25 distinct letters of the alphabet. (Since π is a permutation, knowing the value of π on 25 inputs fully determines the value of π on the last remaining input.) For the Vigen`ere cipher, if the period t is known then the encryption of a plaintext of length t (consecutive) suffices to recover the entire key. If only an upper bound tmax on t is known, then t must be learned as well; this can be done using a single plaintext of length O(t). (Note: it is actually a bit challenging to determine the minimal length of a plaintext that suffices to determine t. We only expect students to understand that a plaintext of length O(tmax) suffices.) 1.6 Assume an attacker knows that a user's password is either abcd or bedg. Say the user encrypts his password using the shift cipher, and the attacker sees the resulting ciphertext. Show how the attacker can determine the user's password, or explain why this is not possible. Solution: In the shift cipher, the relative shift between characters is preserved. Thus an encryption of abcd will always be a ciphertext containing 4 consecutive characters (e.g., lmno), whereas bedg will not. 1.7 Repeat the previous exercise for the Vigen`ere cipher using period 2, using period 3, and using period 4. Introduction 5 Solution: For the sake of clarity, we will mostly omit the fact that operations are modulo 26 in this solution. When the period is 2, it is impossible to determine which password was encrypted. This is due to the fact that the shifts used in the first and third (resp., second and fourth) positions are the same, and the difference between the first and third (resp., second and fourth) characters in the first password is the same as the difference between the first and third (resp., second and fourth) characters in the second password. When the period is 3, it is possible to tell which password was encrypted because the shifts used in the first and fourth positions are the same, but the difference between the first and fourth characters of the first plaintext is not the same as the difference between the first and fourth characters of the second plaintext. When the period is 4, it is impossible to tell which password was encrypted, because using a 4-character key to encrypt a 4-character plaintext is perfectly secret (by analogy to the one-time pad). 1.8 The shift, substitution, and Vigen`ere ciphers can also be defined over the 128-character ASCII alphabet (rather than the 26-character English alphabet). (a) Provide a formal definition of each of these schemes in this case. (b) Discuss how the attacks we have shown in this chapter can be modified to break each of these modified schemes. Solution: We describe the solution for the Vigen`ere cipher, which is the most complex case. (a) Key generation chooses a period t (say, uniform in {1, . . . , tmax } for some specified tmax) and then, for i = 0, . . . , t − 1, chooses uniform ki ∈ {1, . . . , 127}. The encryption of a plaintext m = m1 , . . . , mℓ is c = c1 , . . . , cℓ , where ci := [mi + k[i mod t] mod 128]. Decryption is done in the natural way. (b) The exact same attacks described in the text work, though one must be careful now to let pi be the frequency of the ith ASCII character (so, e.g., the frequency of 'A' is different from the frequency of 'a,' and frequencies of the space character and punctuation is also taken into account). Chapter 2 Perfectly Secret Encryption – Solutions 2.1 Prove that, by redefining the key space, we may assume that the keygeneration algorithm Gen chooses a key uniformly at random from the key space, without changing Pr[C = c | M = m] for any m, c. Solution: If Gen is a randomized algorithm, we may view it as a deterministic algorithm that takes as input a random tape ω of some length; the distribution on the output of Gen is, by definition, the distribution obtained by choosing uniform ω and then running Gen(ω). So, rather than letting the key be the output of Gen, we can simply let the key be ω itself (and redefine the key space accordingly). Formally, given a scheme (Gen, Enc, Dec) in which Gen is randomized, construct a new scheme (Enc′ , Dec′) where the key is a uniform ω. Then define Enc′ω (m) to compute k := Gen(ω) followed by Enck (m), and define decryption analogously. 2.2 Prove that, by redefining the key space, we may assume that Enc is deterministic without changing Pr[C = c | M = m] for any m, c. Solution: As in the previous exercise, if Enc is a randomized algorithm then we may view it as being a deterministic algorithm that also takes a random tape ω as additional input. The distribution on the output of Enck (m) is then, by definition, the distribution obtained by choosing uniform ω and then computing Enck (m; ω). We then define key generation to include ω as well as k (and redefine the key space accordingly). Formally, given a scheme (Gen, Enc, Dec) in which Enc is randomized, construct a new scheme (Gen′ , Enc′ , Dec′ = Dec) as follows. Gen′ computes k ← Gen and also chooses uniform ω; the key is (k, ω). Then define Enc′(k,ω) (m) to be Enck (m; ω). 2.3 Prove or refute: An encryption scheme with message space M is perfectly secret if and only if for every probability distribution over M and every c0 , c1 ∈ C we have Pr[C = c0] = Pr[C = c1]. Solution: This is not true. Consider modifying the one-time pad so encryption appends a bit that is 0 with probability 1/4 and 1 with probability 3/4. This scheme will still be perfectly secret, but ciphertexts ending in 1 are more likely than ciphertexts ending in 0. 7 8 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 2.4 Prove the second direction of Lemma 2.4. Solution: Say (Gen, Enc, Dec) is perfectly secret. Fix two messages m, m′ and a ciphertext c that occurs with nozero probability, and consider the uniform distribution over {m, m′ }. Perfect secrecy implies that Pr[M = m | C = c] = 1/2 = Pr[M = m′ | C = c]. But 1 Pr[C = c | M = m] · Pr[M = m] = Pr[M = m | C = c] = 2 Pr[C = c] 1 Pr[C = c | M = m] = 2 , Pr[C = c] and so Pr[C = c | M = m] = Pr[EncK (m) = c] = Pr[C = c]. Since an analogous calculation holds for m′ as well, we conclude that Pr[EncK (m) = c] = Pr[EncK (m′) = c]. 2.5 Prove Lemma 2.6. Solution: We begin by proving that any encryption scheme that is perfectly secret is perfectly indistinguishable. Every adversary A participating in PrivKeav A,Π defines a fixed pair of plaintext messages m0 , m1 that it outputs in the first step of the experiment. (Note that since A is a deterministic algorithm, it always outputs the same pair of messages.) Fix A and fix m0 , m1 output by A. By Lemma 2.4, for every m0 , m1 ∈ M′ and every c ∈ C, Pr[EncK (m0) = c] = Pr[EncK (m1) = c]. (2.1) In particular, the above holds for m0 , m1 output by A and for any c. Let C0 (resp., C1) denote the set of ciphertexts for which A outputs 0 (resp., 1) at the conclusion of experiment PrivKeav A,Π ; since A is deterministic these sets are well-defined. Note that since A must output either 0 or 1, it follows that C = C0 ∪ C1 . We have: 1 1 · Pr[A outputs 0 | b = 0] + · Pr[A outputs 1 | b = 1] 2 2 1 ∑ 1 ∑ = · Pr[c = EncK (m0)] + · Pr[c = EncK (m1)] 2 2 c∈C0 c∈C1 1 ∑ 1 ∑ = · Pr[c = EncK (m0)] + · Pr[c = EncK (m0)] 2 2 c∈C0 c∈C1 1 ∑ 1 1 = · Pr[c = EncK (m0)] = · (1) = . 2 2 2 Pr[PrivKeav A,Π = 1] = c∈C where the second last equality is due to Equation (2.1). We now proceed to the other direction, that any perfectly indistinguishable encryption scheme is also perfectly secret. We prove the contrapositive. Assume encryption scheme Π is not perfectly secret with respect Perfectly Secret Encryption 9 to Definition 2.1. Then by Lemma 2.4, there must exist two messages m0 , m1 ∈ M and a ciphertext c˜ ∈ C for which Pr[EncK (m0) = c˜] ≠ Pr[EncK (m1) = c˜]. (2.2) Let A be an adversary who outputs m0 , m1 in the first step of PrivKeav A,Π . Then if A receives any ciphertext c′ ≠ c˜ it outputs a random bit b′ ; if it receives the ciphertext c˜, it outputs b′ = 0. Since each message is chosen with probability 1/2 in the experiment, we have Pr[PrivKeav A,Π = 1] 1 1 = · Pr[PrivKeav · Pr[PrivKeav A,Π = 1 | M = m0] + A,Π = 1 | M = m1]. 2 2 Next, Pr[PrivKeav A,Π = 1 | M = m0] = Pr[PrivKeav ˜ = EncK (m0)] A,Π = 1 ∧ c + Pr[PrivKeav ≠ EncK (m0)] A,Π = 1 ∧ c 1 c′ ≠ EncK (m0)] = Pr[˜ c = EncK (m0)] + · Pr[˜ 2 where the second-to-last equality is because A always outputs 0 when receiving c˜ (and when M = m0 this means that PrivKeav A,Π = 1), and the last equality is because A outputs a random bit when given a ciphertext c′ ≠ c˜. A similar analysis for the case that M = m1 (the only difference is that now Pr[PrivKeav ≠ EncK (m1)] = 0) gives A,Π = 1 ∧ c Pr[PrivKeav A,Π = 1 | M = m1] = 1 · Pr[˜ c′ ≠ EncK (m1)]. 2 Putting this all together we have: ( ) 1 1 Pr[PrivKeav = 1] = · Pr[˜ c = Enc (m)] + · Pr[˜ c ≠ Enc (m)] K 0 K 0 A,Π 2 2 1 1 · · Pr[˜ c′ ≠ EncK (m1)] + (2 2) 1 1 = · Pr[˜ c = EncK (m0)] + · (1 − Pr[˜ c = EncK (m0)]) 2 2 1 1 + · · Pr[˜ c′ ≠ EncK (m1)] 2 2 1 1 1 = + · Pr[˜ c = EncK (m0)] + · Pr[˜ c′ ≠ EncK (m1)] 4 4 4 1 1 ≠ + · Pr[˜ c = EncK (m1)] + · Pr[˜ c′ ≠ EncK (m1)] 4 4 4 1 1 1 1 = + = 4 4 2 10 Introduction to Modern Cryptography – 2nd Edition Solutions Manual where the inequality is by Equation (2.2). Since Pr[PrivKeav A,Π = 1] ≠ 1/2 we have that Π is not perfectly indistinguishable. 2.6 For each of the following encryption schemes, state whether the scheme is perfectly secret. Justify your answer in each case. (a) The message space is M = {0, . . . , 4}. Algorithm Gen chooses a uniform key from the key space {0, . . . , 5}. Enck (m) returns [k + m mod 5], and Deck (c) returns [c − k mod 5]. (b) The message space is M = {m ∈ {0, 1}ℓ | the last bit of m is 0}. Gen chooses a uniform key from {0, 1}ℓ−1 . Enck (m) returns ciphertext m ⊕ (k∥0). and Deck (c) returns c ⊕ (k∥0). Solution: (a) The scheme is not perfectly secret. To see this, we can use the equivalent definition of perfect secrecy given by Equation (2.1). If the message is 0, then the ciphertext is 0 if and only if k ∈ {0, 5}. So Pr[EncK (0) = 0] = 1/3. On the other hand, if the message is 1, then the ciphertext is 0 if and only if k = 4. So Pr[EncK (1) = 0] = 1/6 ≠ Pr[EncK (0) = 0]. (b) One can prove that this is perfectly secret by analogy with the onetime pad. (Essentially the final bit of the message is being ignored here, since it is always 0.) 2.7 When using the one-time pad with the key k = 0ℓ , we have Enck (m) = k ⊕ m = m and the message is sent in the clear! It has therefore been suggested to modify the one-time pad by only encrypting with k ≠ 0ℓ (i.e., to have Gen choose k uniformly from the set of nonzero keys of length ℓ). Is this modified scheme still perfectly secret? Explain. Solution: The modified scheme is not perfectly secret. To see this formally, consider the uniform distribution over M = {0, 1}ℓ . For any fixed message α ∈ {0, 1}ℓ , we have Pr[M = α | C = α] = 0 ≠ Pr[M = α]. This contradicts perfect secrecy. We conclude that in order to obtain perfect secrecy, it must be possible to encrypt using the key 0ℓ . This may seem counter-intuitive, since this key does not change the plaintext. However, note that an eavesdropper has no way of knowing if the key is 0ℓ , so the fact that the ciphertext is the same as the plaintext in this case is really of no help to the adversary. Perfectly Secret Encryption 11 2.8 Let Π denote the Vigen`ere cipher where the message space consists of all 3-character strings (over the English alphabet), and the key is generated by first choosing the period t uniformly from {1, 2, 3} and then letting the key be a uniform string of length t. (a) Define A as follows: A outputs m0 = aab and m1 = abb. When given a ciphertext c, it outputs 0 if the first character of c is the same as the second character of c, and outputs 1 otherwise. Compute Pr[PrivKeav A,Π = 1]. (b) Construct and analyze an adversary A′ for which Pr[PrivKeav A′ ,Π = 1] is greater than your answer from part (a). Solution: (a) Say aab is encrypted to give ciphertext c. What is the probability that the first and second characters of c are equal? When t = 1 (which occurs 1/3 of the time) this always happens. But when t ∈ {2, 3} this happens only if the first and second characters of the key are equal, which occurs with probability 1/26. So Pr[A outputs 0 | m0 is encrypted] = 1 2 1 + · ≈ 0.359. 3 3 26 If instead abb is encrypted, then the first and second characters of c can never be equal when t = 1, but are equal with probability 1/26 when t ∈ {2, 3}. Thus, Pr[A outputs 0 | m1 is encrypted] = 2 1 · ≈ 0.026. 3 26 We therefore have Pr[PrivKeav A,Π = 1] 1 = · Pr[A outputs 0 | m0 is encrypted] 2 1 + · Pr[A outputs 1 | m1 is encrypted] 2 1 1 ≈ · 0.359 + · 0.974 ≈ 0.667. 2 2 (b) There are many possible solutions; we present one. Consider the adversary A′ who outputs m0 = aaa and m1 = abc and outputs '0' iff the first and second characters in the ciphertext c are the same, or if the first and last characters in the ciphertext are the same. Call this event E. Say aaa is encrypted. If t ∈ {1, 2} then E always happens. When t = 3 all characters in the ciphertext are uniform and independent; 12 Introduction to Modern Cryptography – 2nd Edition Solutions Manual rather than calculate the probability of E in this case, we just let p denote that probability. Say abc is encrypted. If t = 1 then E never happens. When t = 2 the first and last characters of c are never equal, but the first and second characters are equal with probability 1/26. When t = 3 then all characters in the ciphertext are random and so E occurs with probability p. Putting everything together gives: Pr[PrivKeav A′ ,Π = 1] 1 = · Pr[A′ outputs 0 | m0 is encrypted] 2 1 + · Pr[A′ outputs 1 | m1 is encrypted] 2( ) (( )) 1 2 1 1 1 1 1 1 = · + ·p + · 1− · + ·p ≈ 0.827. 2 3 3 2 3 26 3 2.9 In this exercise, we look at different conditions under which the shift, mono-alphabetic substitution, and Vigen`ere ciphers are perfectly secret: (a) Prove that if only a single character is encrypted, then the shift cipher is perfectly secret. (b) What is the largest message space M for which the mono-alphabetic substitution cipher provides perfect secrecy? (c) Prove that the Vigen`ere cipher using (fixed) period t is perfectly secret when used to encrypt messages of length t. Reconcile this with the attacks shown in the previous chapter. Solution: (a) This can be proved directly (as in the case of the one-time pad) or using Shannon's theorem. (b) Let M be the set of all permutations of the alphabet (i.e., all strings of length 26 with no repeated letter). One can prove directly that the mono-alphabetic substitution cipher is perfectly secret for this message space; we prove it using Shannon's theorem. Briefly, we have |M| = |K| = |C| = 26!, where the size of C follows from the fact that the composition of two permutations is a permutation (and furthermore all permutations can be obtained in this way). This latter fact also proves the second requirement of Shannon's theorem: for every two permutations (namely, plaintext and ciphertext) there exists only one permutation π that maps the plaintext to the ciphertext. This is the largest possible plaintext space, because by Theorem 2.7 we must have |M| ≤ |K| for any perfectly secret encryption scheme. Perfectly Secret Encryption 13 (c) This can be proved directly (as in the case of the one-time pad) or using Shannon's theorem. The attacks in Chapter 1 rely on longer plaintexts being encrypted. 2.10 Prove that a scheme satisfying Definition 2.5 must have |K| ≥ |M| without using Lemma 2.6. Specifically, let Π be an[arbitrary encryption] 1 scheme with |K| < |M|. Show an A for which Pr PrivKeav A,Π = 1 > 2 . Solution: Consider the following A: output uniform m0 , m1 ∈ M. Upon receiving ciphertext c, check (by exhaustive search) whether there exists a key k such that Deck (c) = m0 . If so, output 0; else output 1. When m0 is encrypted then A always outputs 0. On the other hand, when m1 is encrypted then there are at most |K| possible messages that c can decrypt to; since m0 is uniform and independent of m1 the probability that m0 is equal to one of those messages, and so the probability that A outputs 0 in this case, is at most |K|/|M| < 1. We conclude that [ ] 1 1 Pr PrivKeav · Pr[A(EncK (m0)) = 0] + · Pr[A(EncK (m1)) = 1] A,Π = 1 = 2 2 1 1 + · (1 − Pr[A(EncK (m1)) = 0]) > . 2 2 2 2 2.11 Assume we require only that an encryption scheme (Gen, Enc, Dec) with message space M satisfy the following: For all m ∈ M, we have Pr[DecK (EncK (m)) = m] ≥ 2−t . (This probability is taken over choice of the key as well as any randomness used during encryption.) Show that perfect secrecy can be achieved with |K| < |M| when t ≥ 1. Prove a lower bound on the size of K in terms of t. Solution: Let K = {0, 1}ℓ and M = {0, 1}ℓ+t . The key-generation algorithm chooses a uniform string from K. To encrypt a message m ∈ M using key k, let m′ denote the first ℓ bits of m and output c := m′ ⊕ k (both m′ and k have length ℓ). To decrypt a ciphertext c using key k, choose a random string r ← {0, 1}t and output m := (c ⊕ k)∥r. Note that Pr[Deck (Enck (m)) = m] = 2−t because decryption is correct if and only if the random string r chosen during decryption happens to equal the last t bits of m (and this occurs with probability 2−t). Perfect secrecy of this scheme follows from the proof of the one-time pad (indeed, this is exactly a one-time pad on the first ℓ bits of the message). We now prove the following lower bound: THEOREM Let (Gen, Enc, Dec) be a perfectly secret encryption scheme over message space M with Pr[Deck (Enck (m)) = m] ≥ 2−t . Then |K| ≥ |M| · 2−t . 14 Introduction to Modern Cryptography – 2nd Edition Solutions Manual PROOF Let Pr∗ denote the maximum probability with which an unbounded algorithm can cause the stated event to occur. Consider an experiment in which a message m is chosen uniformly from M and we are interested in the maximum probability of correctly guessing m. Clearly, Pr∗ [guess m] = |M|−1 . Now consider an extension of this experiment where a key k is generated using Gen and a ciphertext c ← Enck (m) is computed. Perfect secrecy implies that |M|−1 = Pr∗ [guess m] = Pr∗ [guess m given c]. Continuing, we have Pr∗ [guess m given c] ≥ Pr∗ [guess m and k given c] = Pr∗ [guess k given c] · Pr∗ [guess m given k and c] ≥ |K|−1 · Pr∗ [guess m given k and c] ≥ |K|−1 · 2−t , by the correctness guarantee, and using the fact that m and k are independent. We conclude that |K| ≥ |M| · 2−t . 2.12 Let ε ≥ 0 be a constant. Say an encryption scheme is ε-perfectly secret if for every adversary A it holds that [ ] 1 Pr PrivKeav + ε. A,Π = 1 ≤ 2 (Compare to Definition 2.5.) Show that ε-perfect secrecy can be achieved with |K| < |M| when ε > 0. Prove a lower bound on the size of K in terms of ε. Solution: (See for further details.) We first show that ε-perfect secrecy can be achieved in this case. Let the message space be M = {0, 1}ℓ for some ℓ, and let K ⊂ {0, 1}ℓ be an arbitrary set of size] (1 − ε) · 2ℓ . One can check that for any A [1 we have Pr PrivKeav A,Π = 1 ≤ 2 + ε. One can show that this scheme is optimal, in the sense that any scheme that is ε-perfectly secret must have |K| ≥ (1 − ε) · |M|. 2.13 In this problem we consider definitions of perfect secrecy for the encryption of two messages (using the same key). Here we consider distributions over pairs of messages from the message space M; we let M1 , M2 be random variables denoting the first and second message, respectively. (We stress that these random variables are not assumed to be independent.) We generate a (single) key k, sample a pair of messages (m1 , m2) according to the given distribution, and then compute ciphertexts c1 ← Enck (m1) and c2 ← Enck (m2); this induces a distribution over pairs of ciphertexts and we let C1 , C2 be the corresponding random variables. Perfectly Secret Encryption 15 (a) Say encryption scheme (Gen, Enc, Dec) is perfectly secret for two messages if for all distributions over M × M, all m1 , m2 ∈ M, and all ciphertexts c1 , c2 ∈ C with Pr[C1 = c1 ∧ C2 = c2] > 0: Pr [M1 = m1 ∧ M2 = m2 | C1 = c1 ∧ C2 = c2] = Pr[M1 = m1 ∧ M2 = m2]. Prove that no encryption scheme can satisfy this definition. (b) Say encryption scheme (Gen, Enc, Dec) is perfectly secret for two distinct messages if for all distributions over M × M where the first and second messages are guaranteed to be different (i.e., distributions over pairs of distinct messages), all m1 , m2 ∈ M, and all c1 , c2 ∈ C with Pr[C1 = c1 ∧ C2 = c2] > 0: Pr[M1 = m1 ∧ M2 = m2 | C1 = c1 ∧ C2 = c2] = Pr[M1 = m1 ∧ M2 = m2]. Show an encryption scheme that provably satisfies this definition. Solution: (a) The definition requires the equation to hold for all pairs of plaintexts m, m′ and ciphertexts c, c′ , even when c = c′ but m ≠ m′ . We show that this is impossible. Take the uniform distribution over M and any c such that Pr[C = c ∧ C ′ = c] > 0. Let m, m′ ∈ M be distinct. For any scheme with no decryption error, and any key k, we must have Pr[C = c ∧ C ′ = c | M = m ∧ M ′ = m′ ∧ K = k] = 0. (If not, then decryption of the ciphertext c using the key k gives an error some of the time.) The above implies Pr[C = c ∧ C ′ = c | M = m ∧ M ′ = m′] = 0. So Pr[M = m ∧ M ′ = m′ | C = c ∧ C ′ = c] = 0 but on the other hand Pr[M = m ∧ M ′ = m′] > 0. This holds for any encryption scheme, and so no scheme can satisfy the given definition. (b) Let K be the set of all permutations on M, and let Gen choose a random permutation from K. (Note that Gen is not necessarily efficient; nevertheless, this is allowed in this exercise.) Encryption is carried out by applying the permutation specified by the key to the plaintext, and decryption is carried out by applying the inverse of the permutation to the ciphertext. This scheme satisfies the definition; details omitted. Note: An efficient scheme meeting this definition can be constructed using pairwise-independent permutations (a topic not covered in the book). Chapter 3 Private-Key Encryption – Solutions 3.1 Prove Proposition 3.6. Solution: Let negl1 and negl2 be negligible functions. We prove that: (a) The function negl3 defined by negl3 (n) = negl1 (n) + negl2 (n) is negligible. (b) For any (positive) polynomial p, the function negl4 defined by negl4 (n) = p(n) · negl1 (n) is negligible. We begin with item (1). We show that for every polynomial q(·) there ex1 ists an integer N such that for every n > N it holds that negl3 (n) < q(n) . Fix q(·). Since negl1 and negl2 are negligible, we know that there exist integers N1 and N2 respectively such that for every n > N1 , negl1 (n) < 1 1 2q(n) and for every n > N2 , negl2 (n) < 2q(n) (note that 2q(n) is also a polynomial so the above holds). Take N = max{N1 , N2 }. It follows that 1 1 1 for every n > N , negl3 (n) = negl1 (n) + negl2 (n) < 2q(n) + 2q(n) = q(n) . We conclude that for every polynomial q(·) there exists a integer N such 1 that for every n > N , negl3 (n) < q(n) . We now prove item (2); the proof is very similar so we are brief. Fix q(·); let N be an integer such that for all n > N it holds that negl1 (n) < 1 p(n)·q(n) . This implies that for every n > N it holds that negl4 (n) = p(n) · negl1 (n) < p(n) p(n)·q(n) = 1 q(n) , as required. 3.2 Prove that Definition 3.8 cannot be satisfied if Π can encrypt arbitrarylength messages and the adversary is not restricted to output equallength messages in experiment PrivKeav A,Π . Solution: The encryption algorithm Enc runs in polynomial time. Let p(·) be the polynomial (in the security parameter n) that bounds the running time of Enc when encrypting a single bit. (Thus, Enc runs for at most p(n) steps to encrypt a single bit when the security parameter is 1n .) In particular, the length of the ciphertext generated by Enc when encrypting a single bit is at most p(n) because an algorithm cannot output a string that is longer than its running time. Next, note that the ciphertext generated by encrypting a random string of length p(n) + n has length greater than p(n) with all but negligible probability. (Here we assume, as usual, that decryption is always correct.) 17 18 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Given the above, we can take A to be an adversary that, on input 1n , outputs a pair of plaintexts m0 , m1 in experiment PrivKeav A,Π (n) where m0 is a single bit and m1 is a random string of length p(n) + n. If b = 0 then |c| ≤ p(n), while if b = 1 then |c| > p(n) with all but negligible probability, where c is the challenge ciphertext. Thus, A succeeds with all but negligible probability by outputting b′ = 0 if and only if |c| ≤ p(n). The above holds for any encryption scheme and thus such a definition can never be satisfied. 3.3 Say Π = (Gen, Enc, Dec) is such that for k ∈ {0, 1}n , algorithm Enck is only defined for messages of length at most ℓ(n) (for some polynomial ℓ). Construct a scheme satisfying Definition 3.8 even when the adversary is not restricted to output equal-length messages in experiment PrivKeav A,Π . Solution: To be clear, we are considering a modification of experiment PrivKeav A,Π where the adversary A is not required to output m0 and m1 of the same length, but instead it is only required that m0 and m1 each have length at most ℓ(n). Let Π = (Gen, Enc, Dec) be a scheme that is secure with respect to the original Definition 3.8 (for messages of equal length). Construct a scheme Π′ = (Gen′ , Enc′ , Dec′) as follows: (a) Gen′ is identical to Gen. (b) Upon input a plaintext message m of length at most ℓ = ℓ(n) (where n is the length of the key), Enc′ first sets m′ := 0ℓ−|m| 1∥m and then encrypts m′ using Enc. Note that m′ is always exactly ℓ(n) + 1 bits long. (c) Dec′ applies Dec to the ciphertext, and parses the result as 0t 1∥m for t ≥ 0. It outputs m. It is clear that if Π satisfies Definition 3.8 then Π′ satisfies the modified definition. A complete answer to this exercise requires a proof showing that the existence of an adversary breaking Π′ with respect to the modified definition implies the existence of an adversary breaking Π with respect to Definition 3.8. We describe the reduction informally. Given an adversary A′ who breaks Π′ , we construct an adversary A who takes the pair of plaintexts m0 , m1 output by A′ and pads them in the same way as Enc′ would. Then, it outputs the padded messages to be encrypted. Observe that A outputs equal-length messages, as required. Furthermore, if A′ can correctly guess b with probability non-negligibly greater than 1/2, then A guesses correctly with the same probability. 3.4 Prove the equivalence of Definition 3.8 and Definition 3.9. Private-Key Encryption 19 Solution: Let Π be an encryption scheme. We have: Pr[PrivKeav A,Π (n) = 1] = Pr [output(PrivKeav A,Π (n, b)) = b] b←{0,1} 1 · Pr[output(PrivKeav A,Π (n, 0)) = 0] 2 1 + · Pr[output(PrivKeav A,Π (n, 1)) = 1] 2) 1 (= · 1 − Pr[output(PrivKeav A,Π (n, 0)) = 1] 2 1 + · Pr[output(PrivKeav A,Π (n, 1)) = 1] 2 1 1 (= + · Pr[output(PrivKeav (3.1) A,Π (n, 1)) = 1] 2 2) − Pr[output(PrivKeav A,Π (n, 0)) = 1] . = If Π satisfies Definition 3.9, then there exists a negligible function negl for which 1 1 (+ · Pr[output(PrivKeav A,Π (n, 1)) = 1] 2 2) − Pr[output(PrivKeav A,Π (n, 0)) = 1] 1 1 ≤ + · Pr[output(PrivKeav A,Π (n, 1)) = 1] 2 2 − Pr[output(PrivKeav A,Π (n, 0)) = 1] Pr[PrivKeav A,Π (n) = 1] = ≤ 1 + negl(n); 2 thus, Π satisfies Definition 3.8 also. For the other direction, assume Π satisfies Definition 3.8. By Equation (3.1) we see that there exists a negligible function negl such that negl(n) ≥ Pr[PrivKeav A,Π (n) = 1] − = 1 2 1 (· Pr[output(PrivKeav A,Π (n, 1)) = 1] 2) − Pr[output(PrivKeav A,Π (n, 0)) = 1] . (3.2) Let Aˆ denote an adversary that runs identically to A except that it outputs the complement of whatever A outputs. Then eav Pr[PrivKeav ˆ (n)) = 1] = 1 − Pr[PrivKA,Π (n) = 1] A,Π 1 1 (= + · Pr[output(PrivKeav A,Π (n, 0)) = 1] 2 2) − Pr[output(PrivKeav A,Π (n, 1)) = 1] 20 Introduction to Modern Cryptography – 2nd Edition Solutions Manual and, arguing as above, there exists a negligible function negl′ such that negl′ (n) ≥ Pr[PrivKeav ˆ (n) = 1] − A,Π = 1 2 1 (· Pr[output(PrivKeav A,Π (n, 0)) = 1] 2) − Pr[output(PrivKeav A,Π (n, 1)) = 1] . (3.3) Combining Equations (3.2) and (3.3), we see that Π satisfies Definition 3.9 as well. 3.5 Let |G(s)| = ℓ(|s|) for some ℓ. Consider the following experiment: The PRG indistinguishability experiment PRGA,G (n): (a) A uniform bit b ∈ {0, 1} is chosen. If b = 0 then choose a uniform r ∈ {0, 1}ℓ(n) ; if b = 1 then choose a uniform s ∈ {0, 1}n and set r := G(s). (b) The adversary A is given r, and outputs a bit b′ . (c) The output of the experiment is defined to be 1 if b′ = b, and 0 otherwise. Provide a definition of a pseudorandom generator based on this experiment, and prove that your definition is equivalent to Definition 3.14. (That is, show that G satisfies your definition if and only if it satisfies Definition 3.14.) Solution: The required definition is that G is a pseudorandom generator if for any ppt algorithm A we have Pr[PRGA,G (n) = 1] ≤ 1/2 + negl(n). To see that these definitions are equivalent, note that Pr[PRGA,G (n) = 1] 1 1 = · Prr←{0,1}ℓ(n) [A(r) = 0] + · Prs←{0,1}n [A(G(s)) = 1] 2 2) 1 1 (= + · Prs←{0,1}n [A(G(s)) = 1] − Prr←{0,1}ℓ(n) [A(r) = 1] . 2 2 Thus, if there is an efficient A for which the new definition fails to hold, then Definition 3.14 fails to hold for this A as well. And if there is an efficient attacker A for which Definition 3.14 does not hold, we obtain an efficient attacker for which the new definition does not hold either (possibly by flipping the output of A). 3.6 Let G be a pseudorandom generator with expansion factor ℓ(n) > 2n. In each of the following cases, say whether G′ is necessarily a pseudorandom generator. If yes, give a proof; if not, show a counterexample. Private-Key Encryption 21 def (a) Define G′ (s) = G(s1 · · · s[n/2]), where s = s1 · · · sn . ( ) def (b) Define G′ (s) = G 0|s| ∥s . def (c) Define G′ (s) = G(s) ∥ G(s + 1). Solution: (a) Yes! First, since ℓ(n) > 2n we have that |G′ (s)| > 12 · 2n = n as required for any pseudorandom generator. Let ℓ′ be the expansion factor of G′ ; i.e., ℓ′ is such that |G′ (s)| = ℓ′ (|s|). Fix a probabilistic polynomial-time algorithm D and set def ε(n) = Prr←{0,1}ℓ′ (n) [D(r) = 1] − Prs←{0,1}n [D(G′ (s)) = 1] . By definition of G′ , we have that Prs←{0,1}n [D(G′ (s)) = 1] = Prs←{0,1}n/2 [D(G(s)) = 1], and thus Prr←{0,1}ℓ′ (n) [D(r) = 1] − Prs←{0,1}n/2 [D(G(s)) = 1] = ε(n) = ε′ (n/2), def where ε′ (n) = ε(2n) (note the change in the length of s). Since ε′ must be negligible, we conclude that ε is negligible as well. (b) G′ is not necessarily a pseudorandom generator. To see this, let H : {0, 1}n → {0, 1}4n be a pseudorandom generator, and define G(s) = H(s1 · · · sn/2). As in the previous part, G is a pseudorandom generator. But then G′ (s) = G(0|s| ∥s) = H(0|s|), and clearly G′ is not a pseudorandom generator. Fundamentally, the problem here is that G′ runs G on an input that is not uniformly distributed. (c) G′ is not necessarily a pseudorandom generator. To see this, let H : {0, 1}n → {0, 1}2n be a pseudorandom generator and define G(s) = H(s1 · · · sn−1). As in the previous parts, G is a pseudorandom generator. But then if the last bit of s is 0 we have G′ (s) = G(s)∥G(s + 1) = H(s1 · · · sn−1)∥H(s1 · · · sn−1) (because then s and s + 1 differ only in their final bit), and so with probability 1/2 the two halves of the output of G′ are the same. This is clearly not a pseudorandom generator. Fundamentally, the problem here is that G′ runs G on two correlated (rather than independent) inputs. 22 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 3.7 Prove the converse of Theorem 3.18. Namely, show that if G is not a pseudorandom generator then Construction 3.17 does not have indistinguishable encryptions in the presence of an eavesdropper. Solution: Let D be an efficient algorithm such that def ε(n) = Prr←{0,1}ℓ(n) [D(r) = 1] − Prs←{0,1}n [D(G(s)) = 1] is not negligible. Construct the attacker A that, on input 1n , outputs (m0 , m1) where m0 = 0ℓ(n) and m1 is uniform. Upon receiving ciphertext c, attacker A runs D(c) and outputs the result. Note that when m0 is encrypted, the ciphertext is equal to G(s) for a uniform s, while when m1 is encrypted, the ciphertext is a uniform ℓ-bit string (since m1) is uniform. Thus, Pr[PrivKeav A,Π (n) = 1] ( ) 1 = · Prs←{0,1}n [D(G(s)) = 0] + Prr←{0,1}ℓ(n) [D(r) = 1] 2) 1 1 (= + · Prr←{0,1}ℓ(n) [D(r) = 1] − Prs←{0,1}n [D(G(s)) = 1] , 2 2 which (by assumption) is not negligibly greater than 1/2. 3.8 (a) Define a notion of indistinguishability for the encryption of multiple distinct messages, in which a scheme need not hide whether the same message is encrypted twice. (b) Show that Construction 3.17 does not satisfy your definition. (c) Give a construction of a deterministic (stateless) encryption scheme that satisfies your definition. Solution: (a) The definition is a straightforward modification of Definition 3.19, where experiment PrivKmult A,Π (n) is modified so that all the messages⃗ 1 (but a message may appear in M0 are distinct, and similarly for M⃗ in both M0 and M1). (b) It is easy to see that Construction 3.17 does not satisfy the definition since, for example, when two messages that are identical except for their last bit are encrypted, the result is two ciphertexts that are identical except for their last bit. (c) Let F be a block cipher. Then setting Enck (m) = Fk (m) satisfies the definition even though it is deterministic. 3.9 Prove unconditionally the existence of a pseudorandom function F : {0, 1}∗ × {0, 1}∗ → {0, 1}∗ with ℓkey (n) = n and ℓin (n) = O(log n). Solution: Define keyed function F : {0, 1}n × {0, 1}log n → {0, 1} as follows: Fk (i) outputs the ith bit of k, where the input i is interpreted Private-Key Encryption 23 as an integer in the range {0, . . . , n − 1} and the bits of k are numbered starting at 0. Note that F is exactly implementing a lookup table based on the key k, and so Fk for uniform k is exactly a random function mapping log n-bit inputs to 1-bit outputs. I.e., F is a random function, which is stronger than being pseudorandom. 3.10 Let F be a length-preserving pseudorandom function. For the following constructions of a keyed function F ′ : {0, 1}n × {0, 1}n−1 → {0, 1}2n , state whether F ′ is a pseudorandom function. If yes, prove it; if not, show an attack. def (a) Fk′ (x) = Fk (0∥x) ∥ Fk (1∥x). def (b) Fk′ (x) = Fk (0∥x) ∥ Fk (x∥1). Solution: (a) F ′ is a pseudorandom function. A formal proof is omitted, but relies on the observation that distinct queries to Fk′ result in distinct queries to Fk . (b) F ′ is not a pseudorandom function. To see this, consider querying on the two inputs 0n−1 and 0n−2 1. We have Fk′ (0n−1) = Fk (0n)∥Fk (0n−1 1) and Fk′ (0n−2 1) = Fk (0n−1 1)∥Fk (0n−2 12); note that the second half of Fk′ (0n−1) is equal to the first half of Fk′ (0n−2 1). Formally, define the following attacker A given 1n and access to some function g: Ag (1n): • Query y0 = g(0n−1) and y1 = g(0n−2 1). • Output 1 if and only if the second half of y0 is equal to the first half of y1 . ′ As shown above, we have Prk←{0,1}n [AFk (·) (1n) = 1] = 1. But when g is a random function then y0 and y1 are independent, uniform strings of length 2n, and so the probability that the second half of y0 is equal to the first half of y1 is exactly 2−n . Thus, Prf ←Func [Af (·) (1n) = 1] = 2−n , and the difference ′ Prk←{0,1}n [AFk (·) (1n) = 1] − Prf ←Func [Af (·) (1n) = 1] is not negligible. 24 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 3.11 Assuming the existence of a pseudorandom function, prove that there exists an encryption scheme that has indistinguishable multiple encryptions in the presence of an eavesdropper (i.e., is secure with respect to Definition 3.19), but is not CPA-secure (i.e., is not secure with respect to Definition 3.22). Solution: Let F be a pseudorandom function, and define scheme Π = (Gen, Enc, Dec) as follows: (a) Gen: On input 1n , choose k, s ← {0, 1}n . (b) Enc: On input a key ⟨k, s⟩ and a message m ∈ {0, 1}n , do: i. If m = s output the ciphertext ⟨0, k, s, s⟩. ii. If m ≠ s then choose random r ← {0, 1}n and output the ciphertext ⟨1, s, r, Fk (r) ⊕ m⟩. (c) Dec: On input a key ⟨k, s⟩ and ciphertext c = ⟨b, c1 , c2 , c3 ⟩ do: i. If b = 0 output s. ii. If b = 1 output m := Fk (c2) ⊕ c3 . One can check that decryption always succeeds. It is clear that Π is not secure under a chosen-plaintext attack. This is because an adversary A can query its encryption oracle with input 0n and receive (except with negligible probability) the ciphertext c = ⟨1, s, r, Fk (r)⟩, where s is the second component of the secret key. Then, A can query its encryption oracle again with s and receive back a ciphertext containing k. Given k, of course, A can distinguish encryptions easily. Despite the above, we argue that Π is still secure with respect to Definition 3.18. Intuitively, this is due to the fact that unless one of the plaintexts output by the adversary is equal to s—which occurs with only negligible probability since s is uniform—the scheme is essentially equivalent to Construction 3.30. We omit the formal proof, which is tedious but straightforward. 3.12 Let F be a keyed function and consider the following experiment: The PRF indistinguishability experiment PRFA,F (n): (a) A uniform bit b ∈ {0, 1} is chosen. If b = 1 then choose uniform k ∈ {0, 1}n . (b) A is given 1n for input. If b = 0 then A is given access to a uniform function f ∈ Funcn . If b = 1 then A is instead given access to Fk (·). (c) A outputs a bit b′ . (d) The output of the experiment is defined to be 1 if b′ = b, and 0 otherwise. Private-Key Encryption 25 Define pseudorandom functions using this experiment, and prove that your definition is equivalent to Definition 3.25. Solution: The solution is similar to that of Exercise 3.5, and is omitted. 3.13 Consider the following keyed function F : For security parameter n, the key is an n × n boolean matrix A and an n-bit boolean vector b. Define def FA,b : {0, 1}n → {0, 1}n by FA,b (x) = Ax + b, where all operations are done modulo 2. Show that F is not a pseudorandom function. Solution: Querying FA,b (0n) yields b. Let ei denote the n-bit string with a 1 in position i (and 0s elsewhere). Querying FA,b (ei) yields ai +b, where ai is the ith column of A. This can be easily extended to a fullfledged attack. 3.14 Prove that if F is a length-preserving pseudorandom function, then def G(s) = Fs (1)∥Fs (2)∥ · · · ∥Fs (ℓ) is a pseudorandom generator with expansion factor ℓ · n. Solution: Fix an efficient distinguisher D, and let def ε(n) = Prr←{0,1}ℓn [D(r) = 1] − Prs←{0,1}n [D(G(s)) = 1] . Construct A(1n) as follows: given access to a function g, let yi = g(i) for i = 1, . . . , ℓ. Output D(y1 ∥ · · · ∥yℓ). Clearly Pr[Af (·) (1n) = 1] − Pr[AFk (·) (1n) = 1] = ε(n), and so ε(n) must be negligible. 3.15 Define a notion of perfect secrecy under a chosen-plaintext attack by adapting Definition 3.22. Show that the definition cannot be achieved. Solution: Here we would simply require that for all A (i.e., even unbounded), it holds that Pr[PrivKcpa A,Π (n) = 1] = 1/2. This definition cannot be achieved. To see this, fix an arbitrary scheme Π and let A be the following attacker: request an encryption of 0n , obtaining in return ciphertext c0 . Then output the messages (m0 = 0n , m1 = 1n), and receive in return a ciphertext c. If c = c0 output 0; otherwise, output 1. Note that when m0 is encrypted, there is a nonzero probability that c = c0 , whereas when m1 is encrypted we must have c ≠ c0 . Thus, 1 · (Pr[Enck (m0) = c0] + Pr[Enck (m1) ≠ c0]) 2 > 1/2. Pr[PrivKcpa A,Π (n) = 1] = 3.16 Prove Proposition 3.27. Solution: Intuitively, as long as a distinguisher does not find a collision (namely, a pair x and y for which f (x) = f (y)) the distribution over the 26 Introduction to Modern Cryptography – 2nd Edition Solutions Manual values of a random function and a random permutation are identical. Furthermore, by the analysis of the "birthday problem" (Appendix A.4) we know that a collision is found with only negligible probability as long as only a polynomial number of inputs are queried to f . Thus, a random function and random permutation are computationally indistinguishable. By reduction, the same is true also for a pseudorandom function and pseudorandom permutation. Formally, let f be a uniform length-preserving function and let g denote a uniform length-preserving permutation. Let D be any algorithm that queries its oracle only polynomially many times (any polynomial-time algorithm can only query its oracle only polynomially many times). We claim that there is a negligible function negl such that (3.4) Pr[Df (·) (1n) = 1] − Pr[Dg(·) (1n) = 1] < negl(n) In order to see this, let q(n) be the polynomial bounding the number of queries that D makes to its oracle and assume without loss of generality that D always makes exactly q (distinct) queries. Let coll denote the event that D queries its oracle with two distinct values x and y for which its oracle returns the same reply. Clearly, if D is given access to oracle g then coll occurs with probability zero (because g is a permutation). Furthermore, by Lemma A.15, when D is given access to oracle f , the probability that coll occurs is at most q 2 /2n . (Each query to f on a "fresh" input yields an independent uniform output.) Finally, we observe that conditioned on coll not occurring, the view of D when given oracle f is identical to its view when given oracle g. This is because all oracle responses are independently and uniformly distributed subject only to the constraint that all responses are distinct. Therefore: Pr[Df (·) (1n) = 1] − Pr[Dg(·) (1n) = 1] = Pr[Df (·) (1n) = 1 | coll] · Pr[coll] + Pr[Df (·) (1n) = 1 | coll] · Pr[coll] − Pr[Dg(·) (1n) = 1] = Pr[Df (·) (1n) = 1 | coll] · Pr[coll] + Pr[Dg(·) (1n) = 1] · Pr[coll] − Pr[Dg(·) (1n) = 1] = Pr[Df (·) (1n) = 1 | coll] · Pr[coll] + Pr[Dg(·) (1n) = 1] · (1 − Pr[coll]) − Pr[Dg(·) (1n) = 1] = Pr[Df (·) (1n) = 1 | coll] · Pr[coll] − Pr[Dg(·) (1n) = 1] · Pr[coll] ≤ Pr[coll] ≤ q2 , 2n Private-Key Encryption 27 implying Equation (3.4). Now, let F be a pseudorandom permutation. By the definition, we have that for every probabilistic polynomial-time D there exists a negligible function such that Pr[DFk (·) (1n) = 1] − Pr[Dg(·) (1n) = 1] < negl(n). Combining this with the above, we have that q2 Pr[DFk (·) (1n) = 1] − Pr[Df (·) (1n) = 1] < negl(n) + n . 2 Since q 2 /2n is negligible, we conclude that any pseudorandom permutation F is also a pseudorandom function. 3.17 Assume pseudorandom permutations exist. Show that there exists a function F ′ that is a pseudorandom permutation but is not a strong pseudorandom permutation. Solution: Let F be a length-preserving pseudorandom permutation. Define F ′ as follows: ⌈ n x=k ⟨0 Fk′ (x) = Fk (k) x = Fk−1 (0n) . ⌊ Fk (x) otherwise Fk′ is a permutation for all k, and can be efficiently inverted. We do not prove that F ′ is still a pseudorandom permutation, but intuitively this holds since the only difference between Fk′ and Fk (when an adversary can query in the "forward" direction only) occurs if the adversary queries Fk′ either on the key k or on the point Fk−1 (0n). But the fact that F is pseudorandom means that it is infeasible for an adversary to find either of those two inputs when interacting with Fk′ (·). −1 On the other hand, F ′ is not strongly pseudorandom: since F ′ k (0n) = k, an adversary can learn the key by querying 0n in the reverse direction. Formally, consider the attacker A given access to an oracle g and its inverse g −1 that acts as follows: query g −1 (0n) and obtain a value k. Then choose any x ∉ {k, Fk (k)} and query g(x) to obtain y. Output 1 iff y = Fk (x). It is immediate that Pr[AF ′′ −1 k (·),F k (·) (1n) = 1] = 1. On the other hand, it is easy to see that Pr[Af (·),f −1 (·) (1n) = 1] = 2−n . Thus, F ′ is not strongly pseudorandom. 28 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 3.18 Let F be a pseudorandom permutation, and define a fixed-length encryption scheme (Gen, Enc, Dec) as follows: On input m ∈ {0, 1}n/2 and key k ∈ {0, 1}n , algorithm Enc chooses a random string r ← {0, 1}n/2 of length n/2 and computes c := Fk (r∥m). Show how to decrypt, and prove that this scheme is CPA-secure for messages of length n/2. (If you are looking for a real challenge, prove that this scheme is CCA-secure if F is a strong pseudorandom permutation.) Solution: Decryption is computed by first computing Fk−1 (c) and then outputting the n/2 least significant bits of the result. We now show that this scheme is CPA-secure. In order to show this, we first con˜ that is identical to the above encryption sider an encryption scheme Π scheme, except that a truly random permutation is used instead of a pseudorandom one. Let A be an adversary and let q(·) be a polynomial upper-bounding the running-time of A. We claim that: Pr[PrivKcpa ˜ (n) = 1] ≤ A,Π 1 q(n) + 2 2n/2 Let rc denote the random string used to generate the challenge ciphertext c = Fk (r∥m). There are two cases: (a) The value rc is used by the encryption oracle to answer at least one of A's queries: In this case, A can know which message was encrypted, but the probability of this event occurring is upperbound by q(n)/2n/2 (this is obtained by applying the union bound). (b) The value rc is not used by the encryption oracle to answer any of A's queries: In this case, A learns nothing about the plaintext because the challenge ciphertext is a uniform string (subject to being distinct from all other ciphertexts). Using a similar argument as in the proof of Theorem 3.31 the above equation follows. The rest of the proof follows as in the proof of Theorem 3.31 by showing that the difference when using a pseudorandom permutation instead is at most negligible. We omit a proof of CCA-security. 3.19 Let F be a pseudorandom function, and G a pseudorandom generator with expansion factor ℓ(n) = n + 1. For each of the following encryption schemes, state whether the scheme has indistinguishable encryptions in the presence of an eavesdropper and whether it is CPA-secure. In each case, the shared key is a random k ∈ {0, 1}n . (a) To encrypt m ∈ {0, 1}n+1 , choose uniform r ∈ {0, 1}n and output the ciphertext ⟨r, G(r) ⊕ m⟩. (b) To encrypt m ∈ {0, 1}n , output the ciphertext m ⊕ Fk (0n). Private-Key Encryption 29 (c) To encrypt m ∈ {0, 1}2n , parse m as m1 ∥m2 with |m1 | = |m2 |, then choose uniform r ∈ {0, 1}n and send ⟨r, m1 ⊕Fk (r), m2 ⊕Fk (r+1)⟩. Solution: (a) This scheme does not even have indistinguishable encryptions in the presence of an eavesdropper because the ciphertext doesn't depend on the key. An eavesdropper can easily compute m from c = ⟨r, s⟩ by computing m := G(r) ⊕ s. (b) This scheme has indistinguishable encryptions in the presence of an eavesdropper. To see this, note that Fk (0n) is pseudorandom and so a proof of this fact follows from the proof of Theorem 3.18. The scheme is not CPA-secure because encryption is deterministic. (c) This scheme is CPA-secure. A proof of this is very similar to the proof of Theorem 3.31 except that Repeat denotes the event that r − 1, r or r + 1 is chosen in another ciphertext. A full would include proofs for the above claims. 3.20 Consider a stateful variant of CBC-mode encryption where the sender simply increments the IV by 1 each time a message is encrypted (rather than choosing IV at random each time). Show that the resulting scheme is not CPA-secure. Solution: Define A as follows: (a) Query the encryption oracle with the message m = 0n−1 ∥1. Receive in return a ciphertext ⟨IV, c⟩. (b) If IV is odd (i.e., has low-order bit 1), then output a random bit. (c) If IV is even, then output a pair of messages m0 , m1 where m0 = 0n and m1 is any other message. Receive in return a challenge ciphertext ⟨IV + 1, c′ ⟩. (d) If c′ = c output 0; else output 1. We analyze the success probability of A. If IV is odd, then A succeeds exactly half the time. For any even IV , though, it holds that IV + 1 = IV ⊕ (0n−1 ∥1). Thus, for any even IV we have c = Fk (IV ⊕ m) = Fk (IV ⊕ 0n−1 1 ⊕ m ⊕ 0n−1 1) = Fk ((IV + 1) ⊕ m0). So if m0 is encrypted then c′ = c and A outputs 0, while if m1 is encrypted then c′ ≠ c and A outputs 1. The overall success probability of A is therefore 3/4, and this modified CBC mode is not CPA-secure. 3.21 What is the effect of a single-bit error in the ciphertext when using the CBC, OFB, and CTR modes of operation? 30 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Solution: Say a message m1 , m2 , . . . is encrypted to give a ciphertext c0 , c1 , c2 , . . . , and then a single bit is flipped somewhere in the ciphertext. We look at the effect of decrypting the resulting (modified) ciphertext using each of the stated modes to obtain a message m′1 , m′2 , ... CBC mode. Say a bit is flipped in ci to give modified block c′i . When decrypting, mi is computed as m′i = Fk−1 (c′i) ⊕ c0 and Fk−1 (c′i) will, in general, be completely unrelated to Fk−1 (ci). Thus, m′i will have no relation to mi . The next message block, m′i+1 , is computed as m′i+1 = Fk−1 (ci+1) ⊕ c′i and so m′i+1 will be equal to mi+1 but with a single bit flipped. The rest of the message blocks will be unchanged. OFB mode and CTR mode. In these modes, a bit flip in ci for i > 0 only causes a bit flip in message block mi . However, a bit flip in c0 will (in general) result in all the plaintext blocks being recovered incorrectly. 3.22 What is the effect of a dropped ciphertext block (e.g., if the transmitted ciphertext c1 , c2 , c3 , . . . is received as c1 , c3 , . . .) when using the CBC, OFB, and CTR modes of operation? Solution: Say a message m1 , m2 , . . . is encrypted to give a ciphertext c0 , c1 , c2 , . . . , and then a single block is dropped. We look at the effect of decrypting the resulting (modified) ciphertext using each of the stated modes to obtain a message m′1 , m′2 , ... CBC mode. Say block ci is dropped. The result of decrypting the modified ciphertext is m1 , . . . , mi−1 , m′i+1 , mi+2 , ... I.e., message blocks before position i are recovered correctly, message block i is lost entirely, message block i + 1 is garbled, and message blocks after position i + 1 are recovered correctly (though shifted over by one block). OFB and CTR mode. If ci is dropped then all plaintext blocks before position i are recovered correctly, but from position i and on the remaining plaintext blocks will be incorrect. 3.23 Say CBC-mode encryption is used with a block cipher having a 256-bit key and 128-bit block length to encrypt a 1024-bit message. What is the length of the resulting ciphertext? Solution: The message is 8 = 1024/128 blocks long, so the ciphertext (which includes an IV) is 9 blocks long. Thus, the ciphertext is 1152 bits long. 3.24 Give the details of the proof by reduction for Equation (3.12). Solution: This is exactly analogous to the relevant step in the proof of Theorem 3.31, so is omitted. 3.25 Let F be a pseudorandom function such that for k ∈ {0, 1}n the function Fk maps ℓin (n)-bit inputs to ℓout (n)-bit outputs. Private-Key Encryption 31 (a) Consider implementing CTR-mode encryption using F . For which functions ℓin , ℓout is the resulting encryption scheme CPA-secure? (b) Consider implementing CTR-mode encryption using F , but only for fixed-length messages of length ℓ(n) (which is an integer multiple of ℓout (n)). For which ℓin , ℓout , ℓ does the scheme have indistinguishable encryptions in the presence of an eavesdropper? Solution: (a) In order for counter mode to be secure, it is essential that the counter not repeat even though an arbitrary polynomial number of blocks are encrypted. In order to ensure this, the counter must have superlogarithmic length and so we require ℓin (n) = ω(log n). In contrast, there is no lower or upper bound on the length of ℓout . (b) Consider encryption of a single message of length ℓ. Here, for security to hold we only need to make sure that the counter value does not "wrap" when encrypting. When encrypting a message of length ℓ, we will apply F to ℓ/ℓout counter values. Wrapping occurs if 2ℓin < ℓ/ℓout . So the scheme has indistinguishable encryptions in the presence of an eavesdropper as long as 2ℓin ≥ ℓ/ℓout . 3.26 For any function g : {0, 1}n → {0, 1}n , define g $ (·) to be a probabilistic oracle that, on input 1n , chooses uniform r ∈ {0, 1}n and returns ⟨r, g(r)⟩. A keyed function F is a weak pseudorandom function if for all ppt algorithms D, there exists a negligible function negl such that: $ $ Pr[DFk (·) (1n) = 1] − Pr[Df (·) (1n) = 1] ≤ negl(n), where k ∈ {0, 1}n and f ∈ Funcn are chosen uniformly. (a) Prove that if F is pseudorandom then it is weakly pseudorandom. (b) Let F ′ be a pseudorandom function, and define { def Fk (x) = Fk′ (x) ′ Fk (x + 1) if x is even if x is odd. Prove that F is weakly pseudorandom, but not pseudorandom. (c) Is CTR-mode encryption using a weak pseudorandom function necessarily CPA-secure? Does it necessarily have indistinguishable encryptions in the presence of an eavesdropper? Prove your answers. (d) Prove that Construction 3.30 is CPA-secure if F is a weak pseudorandom function. 32 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Solution: (a) Intuitively, if F cannot be distinguished from a random function even when a distinguisher is allowed to choose the evaluation points, then F cannot be distinguished from a random function when F is evaluated on random points. A formal reduction is straightforward. (b) It is clear that F ′ is not pseudorandom. Specifically, a distinguisher D can query its oracle with an odd x and then with x + 1. If its oracle is F ′ then D will receive back the same value twice. Otherwise, if its oracle is random, it will receive back different values (except with probability 2−n). On the other hand, F ′ is weakly

pseudorandom. Intuitively, this is because the only way to distinguish F′ from a random function is to query it on two consecutive points (as above), but if F′ is evaluated on random inputs then the probability that any two such points happen to be consecutive is negligible. (We omit the formal proof.) (c) Counter mode instantiated with a weak pseudorandom function does not necessarily have indistinguishable encryptions in the presence of an eavesdropper. To see this, take F′ from part (b). When encrypting a message that is at least 3 blocks long, there will be two consecutive plaintext blocks that are XORed with the same value Fk′ (ctr) = Fk′ (ctr + 1). (d) Since a random r is always used in Construction 3.30 as input to F′, the proof that this construction is CPA-secure (even when F is only weakly pseudorandom) is identical. 3.27 Let F be a pseudorandom permutation. Consider the mode of operation in which a uniform value ctr ∈ {0, 1}n is chosen, and the ith ciphertext block ci is computed as ci := Fk (ctr + i + mi). Show that this scheme does not have indistinguishable encryptions in the presence of an eavesdropper. Solution: An encryption of the 2-block message ⟨1⟩, ⟨0⟩ (where ⟨i⟩ is an encoding of the integer i using exactly n bits) has the form ctr, c, c (i.e., the final two ciphertext blocks are equal) whereas an encryption of ⟨0⟩, ⟨1⟩ does not. This can easily be turned into a formal attack. 3.28 Show that the CBC, OFB, and CTR modes of operation do not yield CCA-secure encryption schemes (regardless of F). Solution: CBC mode. Let A be an adversary who outputs a pair of messages m0 = 0n and m1 = 1n, and receives a challenge ciphertext ⟨IV, c⟩. For CBC encryption, we have that c = Fk (IV ⊕ mb). Then A asks for a decryption of ciphertext ⟨0n, c⟩. (Note that IV ≠ 0n except with negligible probability.) The result that A receives back is Fk−1 (Fk (IV ⊕ mb)) ⊕ 0n = IV ⊕ mb. Thus, A knows which plaintext was encrypted. OFB mode. Let A be an adversary who outputs a pair of messages m0 = 0n and m1 = 1n , and receives a challenge ciphertext ⟨IV, c⟩. For OFB encryption we have that c = Fk (IV) ⊕ mb . Adversary A then asks for a decryption of ciphertext ⟨IV, 0n⟩. (Note that c ≠ 0n except with negligible probability.) The result that A receives back is m = Fk (IV) ⊕ 0n . Given this, A computes c ⊕ m; we have that c ⊕ m = mb and so A learns which plaintext was encrypted. CTR mode. The solution is similar to the solution for OFB mode. 3.29 Let Π1 = (Enc1 , Dec1) and Π2 = (Enc2 , Dec2) be two encryption schemes for which it is known that at least one is CPA-secure (but you don't know which one). Show how to construct an encryption scheme Π that is guaranteed to be CPA-secure as long as at least one of Π1 or Π2 is CPA-secure. Provide a full proof of your solution. Solution: Define Π = (Gen, Enc, Dec) as follows: (a) Gen(1n) computes k1 ← Gen1 (1n) and k2 ← Gen2 (1n) and outputs (k1 , k2). (b) Upon input k = (k1 , k2) and m, algorithm Enc chooses a random r ← {0, 1}|m| and outputs c = ⟨Enc1k1 (r), Enc2k2 (r ⊕ m)⟩. (c) Upon input k = (k1 , k2) and c = ⟨c1 , c2 ⟩, compute m1 = Dec1k1 (c1) and m2 = Dec2k2 (c2) and output m1 ⊕ m2 . We prove that the above scheme is CPA-secure as long as either Π1 or Π2 is CPA-secure. Assume that Π1 is CPA-secure (the proof for the case that Π2 is CPA-secure is almost the same). Let A be an adversary attacking Π and let ε be such that Pr[PrivKcpa A,Π (n) = 1] = 1/2 + ε(n). We construct an adversary A1 attacking Π1 as follows. A1 chooses k2 ← Gen2 (1n) and invokes A. For every query m that A makes to its encryption oracle, adversary A1 chooses a random r, queries its oracle with r to receive c1 , computes c2 = Enc2k2 (r ⊕ m), and hands A the ciphertext ⟨c1 , c2 ⟩. When A outputs a pair m0 , m1 , A1 chooses a random r ← {0, 1}|m0| and outputs the pair m0 ⊕ r, m1 ⊕ r. Then, upon receiving back a challenge ciphertext c1 , A1 computes c2 = Enc2k2 (r) and hands A the ciphertext ⟨c1 , c2 ⟩. When A outputs a bit b′, A1 outputs b′ and halts. First, we observe that the view of A in this game by A1 is identical to its view in an execution of experiment PrivKcpa A,Π (n). The experiment in its entirety is not identical because there is a dependence between the two 34 Introduction to Modern Cryptography – 2nd Edition Solutions Manual encryptions in Π. However, since A only sees one of the messages encrypted, and A1 outputs the same bit as A, the success probability is the same. In particular, we have that Pr[PrivKcpa A1 ,Π1 (n) = 1 | b = 0] = Pr[PrivKcpa A,Π (n) = 1 | b = 0] and likewise for b = 1. Therefore, Pr[PrivKcpa A1 ,Π1 (n) = 1] = 1/2 + ε(n) and so by the CPA-security of Π1 we have that ε is negligible, as required. 3.30 Write pseudocode for obtaining the entire plaintext via a padding-oracle attack on CBC-mode encryption using PKCS #5 padding, as described in the text. 3.31 Describe a padding-oracle attack on CTR-mode encryption (assuming PKCS #5 padding is used to pad messages to a multiple of the block length before encrypting). Solution: The high-level idea is the same; the only difference is that now the attacker must modify the bytes of c (rather than ci−1) in order to cause a predictable difference in the ith message block. Chapter 4 Message Authentication Codes – Solutions 4.1 Say Π = (Gen, Mac, Vrfy) is a secure MAC, and for k ∈ {0, 1}n the tag-generation algorithm Mack always outputs tags of length t(n). Prove that t must be super-logarithmic or, equivalently, that if t(n) = O(log n) then Π cannot be a secure MAC. Solution: Assume that t(n) ≤ c log n for some constant c. Consider an adversary A who upon input 1n just outputs an arbitrary m and a uniform t ∈ {0, 1}t(n) . Adversary A succeeds with probability at least 2−t(n) ≥ n−c , which is not negligible. 4.2 Consider an extension of the definition of secure message authentication where the adversary is provided with both a Mac and a Vrfy oracle. (a) Provide a formal definition of security for this case. (b) Assume Π is a deterministic MAC using canonical verification that satisfies Definition 4.2. Prove that Π also satisfies your definition from part (a). Solution: (a) Consider the following modified experiment: The message authentication experiment Mac-forge′A,Π (n): i. Compute k ← Gen(1n). ii. The adversary A is given input 1n and oracle access to Mack (·) and Vrfyk (·, ·). The adversary eventually outputs a pair (m, t). Let Q denote the set of all queries that A asked to its Mack (·) oracle. iii. The output of the experiment is defined to be 1 if and only if (1) Vrfyk (m, t) = 1 and (2) m ∉ Q. The definition of security is the same as Definition 4.2, except that it relates to Mac-forge′ instead of Mac-forge. 35 36 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (b) When Π is a deterministic MAC using canonical verification, every query (m, t) to the Vrfy oracle can be answered by querying Mac(m) and checking whether the result equals t. Thus any adversary using both oracles can be simulated by one using only the Mac oracle, except for verification queries on the message m eventually output, which are handled by guessing. [illegible]

Construction 4.18 instantiated with any CPA-secure encryption scheme and the MAC from the solution to Exercise 4.5. (See also the solution to Exercise 4.21.) ℓ 4.27 Fix ℓ > 0 and a prime p. Let K = Zℓ+1 p , M = Zp , and T = Zp . Define h : K × M → T as [ ] ∑ hk0 ,k1 ,...,kℓ (m1 , . . . , mℓ) = k0 + i ki mi mod p . Prove that h is strongly universal. Solution: Fix distinct m = (m1 , . . . , mℓ) and m′ = (m′1 , . . . , m′ℓ) and arbitrary t, t′ ∈ Zp . Then hk0 ,k1 ,...,kℓ (m) = t and hk0 ,k1 ,...,kℓ (m′) = t′ if and only if ∑ ∑ t = k0 + i ki mi and t′ = k0 + i ki m′i , (4.1) where everything is modulo p. We count the number of keys satisfying the above. Let i∗ be maximal such that mi∗ ≠ m′i∗ . Let k1 , . . . , ki∗ −1 44 Introduction to Modern Cryptography – 2nd Edition Solutions Manual and ki∗ +1 , . . . , kℓ be arbitrary. We then obtain the equations ∑ ∑ t −i≠i∗ ki mi − k0 = ki∗ mi∗ and t′ −i≠i∗ ki m′i − k0 = ki∗ m′i∗ . Subtracting, we obtain ∑ ∑ t − t′ −i≠i∗ ki mi +i≠i∗ ki m′i = ki∗ (mi∗ − m′i∗), which uniquely determines ki∗ since mi∗ − m′i∗ ≠ 0 and p is prime. The original equations then uniquely determine k0 . We conclude that exactly pℓ−1 keys satisfy (4.1), and so the probability that a uniform key satisfies those equations is pℓ−1 /pℓ+1 = 1/|T |2 , as required. 4.28 Fix ℓ, n > 0. Let K = {0, 1}ℓ×n × {0, 1}ℓ (interpreted as a boolean ℓ × n matrix and an ℓ-dimensional vector), let M = {0, 1}n , and let T = {0, 1}ℓ . Define h : K × M → T as hK,v (m) = K · m ⊕ v, where all operations are performed modulo 2. Prove that h is strongly universal. Solution: Fix distinct m, m′ and arbitrary t, t′ . Then hK,v (m) = t and hK,v (m′) = t′ if and only if Km ⊕ v = t and Km′ ⊕ v = t′ . (4.2) Subtracting, we see the above hold only if K · (m − m′) = t − t′ . Let i be the largest index for which the ith coordinate of m−m′ is nonzero. Then we may set all columns of K other than the ith column arbitrarily to any of sℓ·(n−1) possibilities; there is then a unique value of the ith column for which K · (m − m′) = t − t′ . Having fixed K, there is a unique v such that (4.2) hold. We conclude that the number of keys for which hK,v (m) = t and hK,v (m′) = t′ is 2ℓ·(n−1) , and so the probability that a uniform key satisfies those equations is 2ℓ·(n−1) /2ℓ·(n+1) = 1/22ℓ = 1/|T |2 , as required. 4.29 A Toeplitz matrix K is a matrix in which Ki,j = Ki−1,j−1 when i, j > 1; i.e., the values along any diagonal are equal. So an ℓ × n Toeplitz matrix (for ℓ > n) has the form ⎡ ⎤ K1,1 K1,2 K1,3 · · · K1,n ⎢ K2,1 K1,1 K1,2 · · · K1,n−1 ⎥ ⎢ ⎥ ⎢ .. .. ⎥ ⎣ . . . ⎦ . . Kℓ,1 Kℓ−1,1 Kℓ−2,1 · · · Kℓ−n+1,1 Let K = T ℓ×n × {0, 1}ℓ (where T ℓ×n denotes the set of ℓ × n Toeplitz matrices), let M = {0, 1}n , and let T = {0, 1}ℓ . Define h : K × M → T as hK,v (m) = K · m ⊕ v, where all operations are performed modulo 2. Prove that h is strongly universal. What is the advantage here as compared to the construction in the previous exercise? Solution: We first count the number of Toeplitz matrices. In the first row there are n values that may be chosen. In each of the successive Message Authentication Codes 45 (ℓ − 1) rows, the final n − 1 elements are determined by the previous row so there is only one new value to choose. Thus, the total number of Toeplitz matrices is 2n+ℓ−1 , and the size of the key space is 2n+2ℓ−1 . As in the previous exercise, we now count the number of keys for which hK,v (m) = t and hK,v (m′) = t′ . These equations hold only if K(m − m′) = t−t′ . Let i again be the largest index for which the ith coordinate of m − m′ is nonzero. We fill in the entries of K starting from the last row. The first i−1 entries of that row can be set arbitrarily; there is then a unique value for the ith entry such that the equation K(m−m′) = t−t′ can possibly be satisfied. Proceeding to the second-to-last row (we do not yet set the final n−i entries of the last row), we see that the first i−1 entries are already fixed and there is a unique value for the ith entry such that the equation K(m − m′) = t − t′ can possibly be satisfied; note that this determines also the (i + 1)st entry of the bottom row. Continuing in this manner to the top row, we see that all entries of the matrix are uniquely determined except for the final n − i elements of the top row (which in turn determine the final n − i − 1 elements of the second-to-top row, etc.). These can be set arbitrarily. Having fixed K, the original pair of equations then uniquely determine v. We conclude that the number of keys for which hK,v (m) = t and hK,v (m′) = t′ is 2(i−1)+(n−i) = 2n−1 , and so the probability that a uniform key satisfies those equations is 2n−1 /2n+2ℓ−1 = 1/22ℓ = 1/|T |2 , as required. The advantage of this construction is that it has shorter keys. 4.30 Define an appropriate notion of a two-time ε-secure MAC, and give a construction that meets your definition. Solution: A definition is obtained by a straightforward modification of Definition 4.22, where the experiment now allows the attacker to (adaptively) request tags on any two messages of its choice. The natural primitive for constructing a two-time ε-secure MAC is a three-wise independent function, i.e., a keyed function h : K × M → T such that for all distinct m, m′ , m′′ ∈ M and all t, t′ , t′′ ∈ T we have hk (m) = t ∧ hk (m′) = t′ ∧ hk (m′′) = t′′] = 1/|T |3 (where the probability is over choice of k). An example of such a function is given by quadratic polynomials over a field. I.e., let M, T = Zp and K = Z3p , and define hk1 ,k2 ,k3 (m) = k1 m2 + k2 m + k3 . One can check that this satisfies the requirement. 4.31 Let {hn : Kn × {0, 1}10·n → {0, 1}n }n∈N be such that hn is strongly universal for all n, and let F be a pseudorandom function. (When K ∈ Kn we write hK (·) instead of hn,K (·).) Consider the following MAC: Gen(1n) chooses uniform K ∈ Kn and k ∈ {0, 1}n , and outputs (K, k). To authenticate a message m ∈ {0, 1}10·n , choose uniform 46 Introduction to Modern Cryptography – 2nd Edition Solutions Manual r ∈ {0, 1}n and output ⟨r, hK (m) ⊕ Fk (r)⟩. Verification is done in the natural way. Prove that this gives a (computationally) secure MAC for messages of length 10n. Solution: We provide a proof sketch; a full solution would require formal proof. First note that we may, in the standard way, replace Fk by a truly random function f . Let Repeat be the event that, in answering the attacker's Mac queries, an r-value is repeated. This event occurs with negligible probability; we show that as long as it does not occur, the attacker's success probability is 2−n . Assume Repeat does not occur. In this case, note that in each tag ⟨ri , ti ⟩ obtained by the attacker on a message mi , the value ti is uniform and independent of other values, and in particular independent of hK (mi). That is, intuitively, the attacker gets no information about K. Now let m, ⟨r, t⟩ be the candidate forgery output by the attacker. If r was never used to answer any Mac query by the attacker, then f (r) is uniform and independent of anything else and it is immediate that the attacker succeeds only with probability 2−n . Otherwise, since Repeat did not occur, there is a unique Mac query—say, for message mi —for which the tag ⟨r, ti ⟩ was returned. The message/tag pair output by the attacker is valid if and only if hK (m) ⊕ hK (mi) = (t ⊕ f (r)) ⊕ (ti ⊕ f (r)) = t ⊕ ti . But the fact that h is strongly universal implies that the above occurs only with probability 2−n . Chapter 5 Hash Functions and Applications – Solutions 5.1 Provide formal definitions for second preimage resistance and preimage resistance. Prove that any hash function that is collision resistant is second preimage resistant, and any hash function that is second preimage resistant is preimage resistant. Solution: We define two experiments for a hash function Π = (Gen, H), adversary A, and security parameter n. We assume for simplicity that H maps inputs of length 2n to outputs of length n, but this is inessential. DEFINITION Π is preimage resistant if for all ppt adversaries A there is a negligible function negl such that the success probability of A in the following experiment is negligible: (a) A key s is generated by running Gen(1n), and a random x ∈ {0, 1}2n is chosen. (b) The adversary A is given s and H s (x), and outputs x′ . (c) We say that A succeeds if H s (x′) = H s (x). DEFINITION Π is second preimage resistant if for all ppt adversaries A there is a negligible function negl such that the success probability of A in the following experiment is negligible: (a) A key s is generated by running Gen(1n), and a random x ∈ {0, 1}2n is chosen. (b) The adversary A is given s and x, and outputs x′ ≠ x. (c) We say that A succeeds if H s (x′) = H s (x). We first prove that any collision-resistant hash function is also second preimage resistant. Let Π be a collision-resistant hash function. Let A be an adversary and ε a function such that Pr [A(s, x) = x′ ∧ x′ ≠ x ∧ H s (x) = H s (x′)] = ε(n). We construct an adversary A′ that finds a collision in Π as follows. Upon input s, adversary A′ chooses uniform x ∈ {0, 1}2n and runs A(s, x). 47 48 Introduction to Modern Cryptography – 2nd Edition Solutions Manual If A outputs x′ ≠ x such that H s (x) = H s (x′) then A′ outputs the collision (x, x′). It is immediate that A′ outputs a collision—or, stated differently, succeeds in experiment Hash-coll—with probability exactly ε. Thus, ε must be negligible, and so Π is second preimage resistant. We now prove that any second preimage resistant hash function is also preimage resistant. Let Π be a second preimage resistant hash function. Let A be an adversary and set ε(n) = Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x)] . def We construct an adversary A′ that finds a second preimage in Π as follows. Upon input s and x ← {0, 1}2n , A′ invokes A upon (s, H s (x)). If A outputs a value x′ ≠ x such that H s (x′) = H s (x) then A′ outputs x′ and halts. The interesting part of the proof is to show that with high probability A outputs a different preimage x′ ≠ x. Intuitively, this is the case because inputs of length 2n are mapped to outputs of length n. Thus, on average there are many inputs mapped to the same H s (x), and so the probability that x′ = x is small. Formally, we first show that with probability almost one over choice of a random x, there exists at least one other x′ such that H s (x′) = H s (x). We prove this using a simple counting argument. Fix s, and define def T = {x ∈ {0, 1}2n : H s (x) has only one preimage under H s }. Clearly, |T | ≤ 2n (since the range of H s has size 2n). It follows that x ∉ T with probability at least 1 − 2n /22n = 1 − 2−n . If x ∉ T and A outputs a value x′ such that H s (x′) = H s (x), the probability that x′ = x is at most 1/2. (This conclusion follows from the above because the view of A is identical whether x or x′ is chosen.) We conclude that: Pr [A′ (s, x) = x′ ∧ x′ ≠ x ∧ H s (x) = H s (x′)] (5.1) x ≥ Pr [A′ (s, x) = x′ ∧ x′ ≠ x ∧ H s (x) = H s (x′) | x ∉ T] · Pr[x ∉ T] x x = Pr [x′ ≠ x | A(s, H s (x)) = x′ ∧ H s (x) = H s (x′) ∧ x ∉ T] x · Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] · Pr[x ∉ T] x x 1 ≥ · Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] · Pr[x ∉ T]. x 2 x Hash Functions and Applications 49 To complete the proof, note that: Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x)] = Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] · Pr[x ∉ T] + Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∈ T] · Pr[x ∈ T] ≤ Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] · Pr[x ∉ T] + Pr[x ∈ T] ≤ Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] + 2−n . Thus, Pr [A(s, H s (x)) = x′ ∧ H s (x′) = H s (x) | x ∉ T] · Pr[x ∉ T] ≥ ε(n)− Combining this with Equation (5.1), we have Pr [A′ (s, x) = x′ ∧ x′ ≠ x ∧ H s (x) = H s (x′)] ≥ ε(n) − 2−n−1 . 2 Since H is second preimage resistant, ε must be negligible. Because A was arbitrary, this proves that H is preimage resistant. 5.2 Let (Gen1 , H1) and (Gen2 , H2) be two hash functions. Define (Gen, H) so that Gen runs Gen1 and Gen2 to obtain keys s1 and s2 , respectively. Then define H s1 ,s2 (x) = H1s1 (x)‖H2s2 (x). (a) Prove that if at least one of (Gen1 , H1) and (Gen2 , H2) is collision resistant, then (Gen, H) is collision resistant. (b) Determine whether an analogous claim holds for second preimage resistance and preimage resistance, respectively. Prove your answer in each case. Solution: (a) Let A be an adversary that with probability ε(n) outputs a pair x, x′ such that H s1 ,s2 (x) = H s1 ,s2 (x′). In such a case, by the definition of H, we have that H1s1 (x) = H1s1 (x′) and H2s2 (x) = H2s2 (x′). Thus, A finds a collision in both H1 and H2 with probability ε. A full reduction here is straightforward (to attack H1 with key s1 , run Gen2 to get s2 , invoke A with key (s1 , s2), and then output the pair that A outputs). (b) An analogous claim does hold for second preimage resistance. Informally, this is again because a collision in H yields a collision in both H1 and H2 . A formal proof is straightforward. The same does not hold for preimage resistance, i.e., it is possible that (say) H1 is preimage resistant yet H is not. We sketch a ˆ be a preimage resistant counter-example demonstrating this. Let H 50 Introduction to Modern Cryptography – 2nd Edition Solutions Manual hash function mapping n-bit inputs to n/2-bit outputs, and let trunc output the first n/2 bits of its input. Define ˆ s (x2), H1s (x1 ‖x2) = trunc(x1) ‖ H where x1 , x2 ∈ {0, 1}n . Note that H1 is preimage resistant. Let H2 be a keyed function that (for every key s) outputs the last n bits of its input. Then H s1 ,s2 (x1 ‖x2) = H1s1 (x1 ‖x2) ‖ H2s2 (x1 ‖x2) ˆ s1 (x2) ‖ x2 , = trunc(x1) ‖ H and it is trivial to find a preimage in H. ˆ defined 5.3 Let (Gen, H) be a collision-resistant hash function. Is (Gen, H) def s s s ˆ by H (x) = H (H (x)) necessarily collision resistant? ˆ that is, H s (H s (x)) = Solution: Yes. Let x, x′ be a collision for H; s s ′ H (H (x)). There are two cases: (a) H s (x) = H s (x′): in this case, x, x′ is a collision for the original H. Therefore, such a pair can only be found with negligible probability. (b) H s (x) ≠ H s (x′): set y = H s (x) and y ′ = H s (x′). By the assumption in this case, y ≠ y ′ and yet H s (y) = H s (y ′). Thus, y, y ′ is a collision in H. Once again, this implies that such a pair x, x′ can be found with only negligible probability (otherwise, by computing H s once on each one obtains a collision y, y ′ in H s). 5.4 Provide a formal proof of Theorem 5.4 (i.e., describe the reduction). Solution: Let A be a probabilistic polynomial-time adversary and ε a function such that A succeeds in the Hash-coll experiment with (Gen, H) with probability ε. We construct an adversary A′ that succeeds in the Hash-coll experiment with (Gen, h) with probability ε. Upon receiving s, A′ invokes A upon s and receives back a pair x, x′ . If H s (x) ≠ H s (x′) then A′ just halts. Otherwise, A′ uses the strategy described in the portion of the proof of Theorem 5.4 to find a collision in h. Specifically, ′ ′ if L ≠ L′ then A′ outputs zB ‖L and zB ′ ‖L and halts. Otherwise, if ′ ′ L = L , then A works backwards from the last block to the first to find the maximal i∗ . Then, A′ outputs zi∗ −1 ‖xi∗ and z′i∗ −1 ‖x′i∗ and halts. Note that A′ can compute all the z-values by computing H and storing all the intermediate values. By what is shown in the proof in the book, whenever A finds a collision, A′ also finds a collision. Thus, A′ succeeds in Hash-coll for (Gen, h) with probability ε, implying that ε(n) is negligible. Thus, (Gen, H) is collision-resistant as required. 5.5 Generalize the Merkle-Damg˚ard transform (Construction 5.3) for the case when the fixed-length hash function h has input length n + κ (with Hash Functions and Applications 51 κ > 0) and output length n, and the length of the input to H should be encoded as an ℓ-bit value (as discussed in Section 5.3.2). Prove collision resistance of (Gen, H), assuming collision resistance of (Gen, h). Solution: See Construction 5.1-S below. This construction is interesting as long as κ is large enough so that 2κ captures the maximum length of message needed. A proof that this is collision-resistant is similar, though not identical, to the proof of Theorem 5.4. CONSTRUCTION 5.1-S Let (Gen, h) be a fixed-length collision-resistant hash function for inputs of length n + κ and with output length n, where κ > 0. Construct a variable-length hash function (Gen, H) as follows: • Gen: remains unchanged. • H: on input a key s and a string x ∈ {0, 1}∗ of length L < 2κ , do the following: (a) Pad x with zeroes so its length is a multiple of κ. Append L, encoded in binary using exactly κ bits. Parse the resulting string as a sequence of κ-bit blocks X1 , . . . , XB . (b) Set z0 := 0n . (c) For i = 1, . . . , B, compute zi := hs (zi−1 ‖Xi). (d) Output zB . Generalized Merkle-Damg˚ard transform. When κ is small (e.g., κ = 1), a solution is obtained by allocating n additional blocks, always encoding the length using exactly n bits, and appending it to the input before applying the Merkle-Damg˚ard transform. This ensures that if the lengths of two messages are different, then they differ in their last n bits. The proof that this is collision resistant is similar again to the proof of Theorem 5.4. 5.6 For each of the following modifications to the Merkle-Damg˚ard transform (Construction 5.3), determine whether the result is collision resistant. If yes, provide a proof; if not, demonstrate an attack. (a) Modify the construction so that the input length is not included at all (i.e., output zB and not zB+1 = hs (zB ‖L)). (Assume the resulting hash function is only defined for inputs whose length is an integer multiple of the block length.) (b) Modify the construction so that instead of outputting z = hs (zB ‖L), the algorithm outputs zB ‖L. 52 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) Instead of using an IV , just start the computation from x1 . That is, define z1 := x1 and then compute zi := hs (zi−1 ‖xi) for i = 2, . . . , B + 1 and output zB+1 as before. (d) Instead of using a fixed IV , set z0 := L and then compute zi := hs (zi−1 ‖xi) for i = 1, . . . , B and output zB . Solution: (a) This is not collision resistant. In particular, take any x of length that is not an exact multiple of ℓ and then set x′ := x‖0. (Counterexamples are also possible in which there are collisions between two inputs whose lengths are multiples of ℓ, though they rely on a contrived compression function h.) (b) This is collision resistant. The only difference in the proof of Theorem 5.4 is with respect to Case 1. However, here it is even easier because when L ≠ L′ the result cannot be a collision (note that ′ ′ ′ zB ‖L cannot equal zB ′ ‖L when L ≠ L). (c) Likewise, this is collision resistant and the proof is the same. (d) This is not necessarily secure. Assume there is a collision-resistant compression function such that it is possible to efficiently find some L, x ∈ {0, 1}n such that hs (L, x) = L − n for all s). (A full answer would show that it is possible to construct a hash function with this property, assuming collision-resistant hash functions exist.) In this case, the hash of any message y of length L − 1 is equal to the hash of the message (x, y) of length L. 5.7 Assume collision-resistant hash functions exist. Show a construction of a fixed-length hash function (Gen, h) that is not collision resistant, but such that the hash function (Gen, H) obtained from the Merkle-Damg˚ard transform to (Gen, h) as in Construction 5.3 is collision resistant. ˆ : {0, 1}2n−1 → {0, 1}n−1 be a collision-resistant comSolution: Let h pression function, and define h : {0, 1}2n → {0, 1}n such that ˆ s (x) and hs (1‖x) = 1n hs (0‖x) = 0‖h (where x ∈ {0, 1}2n−1). Clearly (Gen, h) is not collision resistant. But one can show that the hash function (Gen, H) obtained from applying the Merkle-Damg˚ard transform to (Gen, h) is collision-resistant. 5.8 Prove or disprove: if (Gen, h) is preimage resistant, then so is the hash function (Gen, H) obtained by applying the Merkle-Damg˚ard transform to (Gen, h) as in Construction 5.3. Solution: (Note: for preimage-resistance to be defined we must specify some input length. We consider inputs of length L, where L is arbitrary.) This is, in general, false. To see this, let (Gen, h) be preimage Hash Functions and Applications 53 resistant but such that hs (x‖L) = 0n for all s and all x ∈ {0, 1}n . (A full answer would prove that such a function exists assuming preimageresistant hash functions exist at all.) Then the Merkle-Damg˚ard transform applied to (Gen, h) yields a hash function (Gen, H) for which every L-bit input hashes to 0n (and so is clearly not preimage-resistant). 5.9 Prove or disprove: if (Gen, h) is second preimage resistant, then so is the hash function (Gen, H) obtained by applying the Merkle-Damg˚ard transform to (Gen, h) as in Construction 5.3. Solution: This is, in general, false. To see this, let (Gen, h) be second preimage resistant but with the property that, for any x ∈ {0, 1}n , hs ([s]n ‖x) = hs (x‖[s]n) = 0n and hs (02n) = [s]n , where [s]n denotes the first n bits of s. (A full answer would prove that such a hash function exists assuming second preimage-resistant hash functions exist at all.) Then the Merkle-Damg˚ard transform applied to (Gen, h) yields a hash function (Gen, H) for which there is the following attacker finding a second preimage: (a) Output 0n . (b) Obtain key s. (c) Output [s]n . This attacker succeeds with probability 1 since H s (0n) = hs (hs (0n , 0n), ⟨n⟩) = hs ([s]n , ⟨n⟩) = 0n and H s ([s]n) = hs (hs (0n , [s]n), ⟨n⟩) = hs ([s]n , ⟨n⟩) = 0n , where ⟨n⟩ denotes an n-bit encoding of the integer n. 5.10 Before HMAC, it was common to define a MAC for arbitrary-length messages by Macs,k (m) = H s (k‖m) where H is a collision-resistant hash function. (a) Show that this is never a secure MAC when H is constructed via the Merkle-Damg˚ard transform. (Assume the hash key s is known to the attacker, and only k is kept secret.) (b) Prove that this is a secure MAC if H is modeled as a random oracle. Solution: (a) (For simplicity we omit s.) Let ⟨i⟩ denote an n-bit encoding of the integer i. We demonstrate an attack: first request a tag for an arbitrary message m ∈ {0, 1}n and receive in return a tag t. Then compute t′ = h(t, ⟨3n⟩), and output the forged tag t′ on the 3block message m, t, ⟨2n⟩. This succeeds with probability 1 since t = h(h(h(0n , k), m), ⟨2n⟩) so then t′ = h(h(h(h(0n , k), m), ⟨2n⟩), ⟨3n⟩). 54 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (b) While security (when H is modeled as a random oracle) can be proven directly, a proof also follows along the lines of the proof of Theorem 4.6 using the result of the following exercise. We therefore omit further details. 5.11 Prove that the construction of a pseudorandom function given in Section 5.5.1 is secure in the random-oracle model. Solution: Let us be clear about the probability spaces of the experiments we will be considering. Let the security parameter be n. In the first experiment, a random function H : {0, 1}2n → {0, 1}n and a random key k ∈ {0, 1}n are chosen, and A is given access to H as well def as the function Fk (·) = H(k‖·). In the second experiment, a random function H : {0, 1}2n → {0, 1}n and an independent random function f : {0, 1}n → {0, 1}n are chosen, and A is given access to H and f . We want to show that the following is negligible: Pr[AH(·),H(k‖·) (1n) = 1] − Pr[AH(·),f (·) (1n) = 1] . Observe that the views of A in the two experiments are identical (i.e., A gets uniform and independent responses from each of its oracles) until A make a query of the form k‖⋆ to its H-oracle in the first experiment. Since A has no information about k, the probability that it ever makes such a query is q/2n , where q is a bound on the total number of queries made by A. Since q must be polynomial, this occurs with negligible probability. 5.12 Prove Theorem 5.11. Solution: The proof is similar to the proof of Theorem 5.4; namely, we show that a collision in MT t implies a collision in H. (For simplicity, we leave the key s implicit.) Consider two inputs (x1 , . . . , xt) ≠ (x′1 , . . . , x′t) for which MT t (x1 , . . . , xt) = h = MT t (x′1 , . . . , x′t). Following Figure 5.5, we can imagine a pair of binary trees with 1+log t levels where h is at the 0th level of both trees and the inputs (x1 , . . . , xt in one case and x′1 , . . . , x′t in the other case) are at the tth level of the trees. Let i denote the least integer for which the values at level i are equal in the two trees but the values at level i + 1 are not. Then there is a collision in H at that level. 5.13 Show how to find a collision in the Merkle tree construction if t is not fixed. Specifically, show how to find two sets of inputs x1 , . . . , xt and x′1 , . . . , x′2t such that MT t (x1 , . . . , xt) = MT 2t (x′1 , . . . , x′2t). Solution: We give a solution for t = 2 for concreteness. Let x′1 , x′2 , x′3 , x′4 be arbitrary. Set x1 = H(x′1 , x′2) and x2 = H(x′3 , x′4). Then notice that MT 2 (x1 , x2) = MT 4 (x′1 , . . . , x′4). Hash Functions and Applications 55 5.14 Consider the scenario introduced in Section 5.6.2 in which a client stores files on a server and wants to verify that files are returned unmodified. (a) Provide a formal definition of security for this setting. (b) Formalize the construction based on Merkle trees as discussed in Section 5.6.2. (c) Prove that your construction is secure relative to your definition under the assumption that (GenH , H) is collision resistant. Solution: (a) We first define the semantics of a scheme (we note that our formulation is not the only one possible). Let Π = (Upload, Return, Vrfy) be a triple of algorithms that work as follows: • Upload takes as input files x1 , . . . , xt and returns state s. • Return takes as input an index i and files x1 , . . . , xt , and returns a proof πi . • Vrfy takes as input state s, an index i, a file x, and a proof π and outputs a bit indicating acceptance or rejection. In intended operation of the scheme, the client begins by computing s ← Upload(x1 , . . . , xt), sending x1 , . . . , xt to the server, and storing s locally. When the client wants file i, it sends i to the server who then computes π ← Return(i, x1 , . . . , xt) and sends back xi and π. The client then computes Vrfy(s, i, xi , π). Correctness requires that, when run as described, Vrfy returns 1. We now define security. (Again, other definitions are possible.) DEFINITION Π is secure if for all ppt adversaries A there is a negligible function negl such that the success probability of A in the following experiment is negligible: i. A outputs x1 , . . . , xt , i, x, and π. ii. Compute s ← Upload(x1 , . . . , xt). iii. A succeeds if Vrfy(s, i, x, π) = 1 and x ≠ xi . (b) • Upload(x1 , . . . , xt) computes h := MT t (x1 , . . . , xt) and lets s := (h, t). • Return(i, x1 , . . . , xt) computes MT t (x1 , . . . , xt) and lets π be the values of the nodes in the Merkle tree adjacent to the path from xi to the root (as described in Section 5.6.2). • Vrfy(s, i, x, π), where s = (h, t), uses t, π, x, and i to recompute a candidate value h′ for the root of the Merkle tree (as described in Section 5.6.2). It outputs 1 iff h′ = h. 56 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) A proof of security follows by showing that if A succeeds in the experiment above, a collision in H is found. We omit the details. 5.15 Prove that the commitment scheme discussed in Section 5.6.5 is secure in the random-oracle model. Solution: Binding follows immediately from the fact that H is collisionresistant (as discussed in Section 5.5.1). For hiding, we provide only a sketch of the proof. Consider an attacker A in experiment HidingA,Com (n) that makes q queries to H. Let r denote the random value used by the sender, and note that (1) the view of A is independent of b (i.e., independent of which message is chosen) unless A makes a query of the form H(⋆‖r) (since then both the commitment as well as all answers from the random oracle are uniform and independent); (2) the probability that A makes such a query is at most q/2n . Chapter 6 Practical Constructions of Symmetric-Key Primitives – Solutions 6.1 Assume a degree-6 LFSR with c0 = c5 = 1 and c1 = c2 = c3 = c4 = 0. (a) What are the first 10 bits output by this LFSR if it starts in initial state (1, 1, 1, 1, 1, 1)? (b) Is this LFSR maximal length? Solution: (a) 1, 1, 1, 1, 1, 1, 0, 1, 0, 1 (b) Yes, one can check that this LFSR has maximal length. 6.2 In this question we consider a nonlinear combination generator, where we have a degree-n LFSR but the output at each time step is not s0 but instead g(s0 , . . . , sn−1) for some nonlinear function g. Assume the feedback coefficients of the LFSR are known, but its initial state is not. Show that each of the following choices of g does not yield a good pseudorandom generator: (a) g(s0 , . . . , sn−1) = s0 ∧ s1 . (b) g(s0 , . . . , sn−1) = (s0 ∧ s1) ⊕ s2 . Solution: (a) The first bit of the output is 0 with probability 3/4. This easily leads to a distinguisher. (0) (0) (b) Let s0 , . . . , sn−1 be the initial state, and denote by y1 , . . . , yn the (0) first n bits of output. If s1 = 0 (which occurs with probabil(0) (0) (0) (0) ity 1/2) then y1 = (s0 ∧ s1) ⊕ s2 = s2 . Furthermore, in the (1) (0) (1) (0) next step, we have s0 = s1 = 0 and so y2 = s2 = s3 . At this (0) (0) (2) (2) point, the attacker knows s2 and s3 which equal s0 and s1 , (2) (2) (2) respectively. But y3 = (s0 ∧ s1) ⊕ s2 , and so from y3 the at(2) (0) tacker can compute s2 = s4 . Continuing in the same way, the (0) (0) attacker can compute s2 , . . . , sn−1 from y1 , . . . , yn . 57 58 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Using the above observation, the attacker works as follows. It (0) (0) (0) assumes that s1 = 0 and computes values s2 , . . . , sn−1 based on this assumption. Then, for each of the two possible values of (0) (0) s0 ∈ {0, 1} and under the assumption that s1 = 0, it computes two possibilities for the next n output bits of the generator. If either of these is correct, the attacker outputs 1; otherwise, it outputs 0. An easy calculation shows that the attacker outputs 1 with probability at least 1/4 when receiving the output of the generator, but outputs 1 with probability at most 2/2n when receiving a uniform string. This is a distinguishing attack. In reality the attack is much more severe since with probability 1/2, the attacker learns the entire initial seed. 6.3 Let F be a block cipher with n-bit key length and block length. Say there is a key-recovery attack on F that succeeds with probability 1 using n chosen plaintexts and minimal computational effort. Prove formally that F cannot be a pseudorandom permutation. Solution: A distinguisher D runs the strategy for learning k based on its oracle queries. Then, D asks a new query x (that it did not query ? previously) and receives back the oracle response y. If y = y ′ , where def y ′ = Fk (x), then D outputs 1; otherwise it outputs 0. By the assumption of the question Pr[DFk (·) (1n) = 1] = 1. In contrast, we claim that when D has access to a random permutation: Pr[Df (·) (1n) = 1] ≤ 1 . 2n − n This holds because given k and x, the value y ′ = Fk (x) is fixed. There are then two cases: either y ′ ∈ {f (x1), . . . , f (xn)}, where x1 , . . . , xn are the n queries made by D to its oracle in the first phase of its execution, or not. In the first case, f (x) cannot possibly be equal to y ′ (since f is a permutation). In the second case, the probability that f (x) = y ′ is exactly 1/(2n − n). We conclude that 1 , Pr[DFk (·) (1n) = 1] − Pr[Df (·) (1n) = 1] ≥ 1 − n 2 −n and F is therefore not a pseudorandom permutation. 6.4 In our attack on a one-round SPN, we considered a block length of 64 bits and 8 S-boxes that each take a 8-bit input. Repeat the analysis for the case of 16 S-boxes, each taking a 4-bit input. What is the complexity of the attack now? Repeat the analysis again with a 128-bit block length and 16 S-boxes that each take an 8-bit input. Practical Constructions of Symmetric-Key Primitives 59 Solution: For the case of 16 S-boxes each taking a 4-bit input, we have that for each guess of 4 bits of k2 there is exactly one possibility for k1 . The attacker therefore prepares 16 lists, each of length 24 . The overall cost is thus 16 · 24 = 28 (less than for the case studied in the text, as is to be expected). Next for the case of a 128-bit block length and 16 S-boxes taking 8-bit inputs, the analysis is exactly the same as in the text except that now there are 16 lists instead of 8 lists. Thus, the complexity is 16 · 28 = 212 (instead of 211 for the case in the text). Observe that the size of the S-box is much more important than the size of the block. 6.5 Consider a modified SPN where instead of carrying out the key-mixing, substitution, and permutation steps in alternating order for r (full) rounds, the cipher instead first applies r rounds of key mixing, then carries out r rounds of substitution, and finally applies r mixing permutations. Analyze the security of this construction. Solution: This is equivalent to a single-round of a substitution-permutation network. To see this, note that r rounds of key-mixing are equivalent to one round of key-mixing (since XORing the r keys k1 , . . . , kr is equivalent to XORing the single key k1 ⊕· · ·⊕kr). Similarly, r rounds of substitution are equivalent to one round of substitution, and applying r mixing permutations sequentially is equivalent to applying a single mixing permutation. 6.6 In this question we assume a two-round SPN with 64-bit block length. (a) Assume independent 64-bit sub-keys are used in each round, so the master key is 192 bits long. Show a key-recovery attack using much less than 2192 time. (b) Assume the first and third sub-keys are equal, and the second subkey is independent, so the master key is 128 bits long. Show a key-recovery attack using much less than 2128 time. Solution: Since this question does not mention the size of the S-boxes, we describe attacks that work independently of the S-box size. (a) If an attacker guesses the first two sub-keys k1 , k2 , then it can compute the first two rounds of the SPN and is left only with the sub-key mixing of k3 . Now, given a plaintext/ciphertext pair (x, y), it can carry out this computation forward from x using its guess of k1 , k2 . Then, k3 will equal the XOR of the result of the first two rounds with y. Thus, the attacker can prepare a list of 2128 candidates for k1 , k2 , k3 . Since the size of the block is 64 bits, an incorrect key will map a plaintext x to its correct ciphertext y 60 Introduction to Modern Cryptography – 2nd Edition Solutions Manual with probability approximately 2−64 . Thus, an incorrect key will correctly map 3 given plaintexts to their given ciphertexts with probability only (2−64)3 = 2−192 . Since there are 2128 candidates, only 1 will remain after 3 more plaintext/ciphertext pairs, except with probability 2−64 . The complexity of the attack is therefore 4 · 2128 = 2130 time and 2128 memory. (b) Denote the first and third sub-keys by k1 , and the second subkey by k2 . Let (x, y) be a plaintext/ciphertext pair. For every possible guess of k1 , the attacker can compute the SPN forward from x to before the mixing of sub-key k2 (this involves mixing k1 which is "known" by the guess, S-box mixing and permutation). Likewise, the attacker can invert the SPN from y up to just after the mixing of sub-key k2 . Since the third sub-key is also k1 , this is also "known". Then, given the values before and after the mixing of k2 , we have that k2 is just the XOR of these values. Thus, the attacker can prepare a list of 264 candidates of (k1 , k2) in time 264 . These can then be checked on another two ciphertexts and with high probability only one will remain (as in the previous solution). 6.7 What is the output of an r-round Feistel network when the input is (L0 , R0) in each of the following two cases: (a) Each round function outputs all 0s, regardless of the input. (b) Each round function is the identity function. Solution: (a) Using the formula in Equation (6.3) we have that L1 = R0 and R1 = L0 ⊕ f1 (R0) = L0 ⊕ 0n = L0 . Similarly, L2 = R1 = L0 and R2 = L1 = R0 . Thus, if r is even then the output is (L0 , R0), and if r is odd then the output is (R0 , L0). (b) Again, using the formula in Equation (6.3), we have that L1 = R0 and R1 = L0 ⊕ f1 (R0) = L0 ⊕ R0 . Then, L2 = R1 = L0 ⊕ R0 and R2 = L1 ⊕R1 = R0 ⊕L0 ⊕ R0 = L0 . Continuing for one more round, we have L3 = R2 = L0 and R3 = L2 ⊕ R2 = L0 ⊕ R0 ⊕ L0 = R0 , bringing us back to the beginning. We conclude that if r is a multiple of 3, then the output equals (L0 , R0). If r = 1 mod 3 then the output equals (R0 , L0 ⊕ R0), and if r = 2 mod 3 then the output equals (L0 ⊕ R0 , L0). 6.8 Let Feistelf1 ,f2 (·) denote a two-round Feistel network using functions f1 and f2 (in that order). Show that if Feistelf1 ,f2 (L0 , R0) = (L2 , R2), then Feistelf2 ,f1 (R2 , L2) = (R0 , L0). Practical Constructions of Symmetric-Key Primitives 61 Solution: The formula for computing Feistelf1 ,f2 (L0 , R0) is: L1 = R0 R1 = L0 ⊕ f1 (R0) L2 = R1 = L0 ⊕ f1 (R0) R2 = L1 ⊕ f2 (R1) = R0 ⊕ f2 (L2). (6.1) (6.2) Now, taking this pair (L2 , R2), we compute Feistelf2 ,f1 (R2 , L2). We denote L′0 = R2 and R0′ = L2 and the intermediate values computed by L′1 , R1′ and L′2 , R2′ . Using the same formula as above, we have: L′1 = R0′ R1′ = L′0 ⊕ f2 (R0′) L′2 = L′0 ⊕ f2 (R0′) R2′ = L′1 ⊕ f1 (R1′) = R0′ ⊕ f1 (L′2) (note that f2 and f1 have been reversed). Plugging in L′0 = R2 and R0′ = L2 we have that L′2 = R2 ⊕ f2 (L2) and R2′ = L2 ⊕ f1 (L′2). Using Equation (6.2), we have that R2 ⊕f2 (L2) = R0 and thus L′2 = R0 . Thus, R2′ = L2 ⊕ f1 (L′2) = L2 ⊕ f1 (R0). However, by Equation (6.1), we have that L2 ⊕ f1 (R0) = L0 and so R2′ = L0 . We conclude that Feistelf2 ,f1 (R2 , L2) = (R0 , L0), as required. 6.9 For this exercise, rely on the description of DES given in this chapter, but use the fact that in the actual construction of DES the two halves of the output of the final round of the Feistel network are swapped. That is, if the output of the final round of the Feistel network is (L16 , R16), then the output of DES is (R16 , L16). (a) Show that the only difference between computation of DESk and DESk−1 is the order in which sub-keys are used. (Rely on the previous exercise.) (b) Show that for k = 056 it holds that DESk (DESk (x)) = x. (c) Find three other DES keys with the same property. These keys are known as weak keys for DES. (Note: the keys you find will differ from the actual weak keys of DES because of differences in our presentation.) (d) Do these 4 weak keys represent a serious vulnerability in the use of triple-DES as a pseudorandom permutation? Explain. Solution: 62 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (a) Denote by Feistel′f1 ,f2 a modified two-round Feistel network using round functions f1 and f2 in that order, where the modification is that the two halves of the output of the final round are swapped. In the previous exercise we showed that if y = Feistel′f1 ,f2 (x), then x = Feistel′f2 ,f1 (y). (This is not true for the "regular" Feistel network, where the two halves of the output of the final round are not swapped.) Now consider the computation Feistel′f2 ,f1 on input (L′0 , R0′) = (R2 , L2). We call the results of the first round L′1 and R1′ , and the results of the second round L′2 and R2′ . Then L′1 = R0′ = R2 = L0 ⊕ f1 (R0) and R1′ = L′0 ⊕ f2 (R0′) = R0 . Next, L′2 = R1′ = L1 = R0 , and R2′ = L′1 ⊕f1 (R1′) = L0 ⊕f1 (R0)⊕f1 (L1) = L0 ⊕f1 (R0)⊕f1 (R0) = L0 . Swapping the result, we obtain output (L0 , R0) which is the original input we started with. The extension to the 16 rounds of DES follows by applying the above argument 8 times. (b) When k = 056 , all sub-keys in all rounds are exactly the same (every sub-key is an all-0 string). By the previous exercise, this means that decryption and encryption are identical (because all sub-keys are the same). Thus, DES056 (DES056 (x)) = DES0−1 56 (DES056 (x)) = x. The use of such a key is a vulnerability because it means that an adversary can decrypt a challenge ciphertext using a chosenplaintext attack. (c) Three other keys with this property are 156 , 028 ‖128 and 128 ‖028 . The two latter keys are also weak keys because, as described in the book, the sub-keys are derived by taking subsets of the left and right halves of the key in each round. Thus, in all rounds we have the sub-key 024 ‖124 (or 124 ‖024) and so once again encryption and decryption are the same. (We remark that the actual weak keys in DES are different because the description in the book is a simplification of DES. Specifically, the two halves of the master key that are used for deriving the sub-keys are not the left and right halves of the master key.) (d) The existence of these keys does not constitute any significant problem for DES. The reason for this is that the probability that one of these keys is chosen is 4/256 = 2−54 . 6.10 Show that DES has the property that DESk (x) = DESk¯ (¯ x) for every key k and input x (where z¯ denotes the bitwise complement of z). (This is called the complementarity property of DES.) Does this represent a serious vulnerability in the use of triple-DES as a pseudorandom permutation? Explain. Practical Constructions of Symmetric-Key Primitives 63 Solution: Let fˆ be the DES mangler function. We first claim that for ¯ x every key k and every input x, it holds that fˆ(k, x) = fˆ(k, ¯). To see this, notice that for input x and key k, the input to the S-boxes equals E(x) ⊕ k where E is the expansion function. Since E simply duplicates half of the bits of its input, we have that E(¯ x) equals E(x). Therefore E(¯ x) ⊕ k¯ = E(x) ⊕ k¯ = E(x) ⊕ k. Since the input to the S-boxes is the same, the output from the S-boxes is also the same. Applying the mixing permutation does not change the fact that the outputs are equal. ¯ x We conclude that fˆ(k, x) = fˆ(k, ¯). Next look at the entire Feistel structure. For input L0 , R0 and key k, the values after the first round are L1 = R0 and R1 = L0 ⊕ fˆ(k1 , R0). By the above, for input L0 , R0 and key k¯ the values after the first round are L′1 = R0 = L1 and R1′ = L0 ⊕ fˆ(k¯1 , R0) = L0 ⊕ fˆ(k1 , R0) = L0 ⊕ fˆ(k1 , R0) = R1 . Since the above continues to hold after each round of the Feistel network is applied, we see that DESk (x) = DESk¯ (¯ x). 6.11 Describe attacks on the following modifications to DES: (a) Each sub-key is 32 bits long, and the round function simply XORs the sub-key with the input to the round (i.e., fˆ(k, R) = ki ⊕ R). For this question, the key schedule is unimportant and you can treat the sub-keys ki as independent keys. (b) Instead of using different sub-keys in every round, the same 48-bit sub-key is used in every round. Show how to distinguish the cipher from a random permutation in ≪ 248 time. Solution: (a) Let us follow a few rounds of the computation, where ki denotes the round-i sub-key. Starting with (L0 , R0) we get L1 = R0 and R1 = L0 ⊕ fˆ(k1 , R0) = L0 ⊕ R0 ⊕ k1 . Then L2 = L0 ⊕ R0 ⊕ k1 and R2 = R0 ⊕ L0 ⊕ R0 ⊕ k1 ⊕ k2 = L0 ⊕ k1 ⊕ k2 . Continuing, we see that the final output is a simple linear function of L0 , R0 , and the sub-keys. It can thus be distinguished easily from a random permutation. (b) By the result of Exercise 6.9, we know that when all sub-keys are the same encryption and decryption are identical. It is therefore trivial to distinguish this cipher from a random permutation. 64 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 6.12 (This exercise relies on Exercise 6.9.) Our goal is to show that for any weak key k of DES, it is easy to find an input x such that DESk (x) = x. (a) Assume we evaluate DESk on input (L0 , R0), and the output after 8 rounds of the Feistel network is (L8 , R8) with L8 = R8 . Show that the output of DESk (L0 , R0) is (L0 , R0). (Recall from Exercise 6.9 that DES swaps the two halves of the 16th round of the Feistel network before outputting the result.) (b) Show how to find an input (L0 , R0) with the property in part (a). Solution: (a) By Exercise 6.9(a), the only difference between DESk and DESk−1 is the order in which sub-keys are used. When k is a DES weak key, it follows that all round functions are the same. For this reason, DESk (DESk (x)) = x as shown in Exercise 6.9(b). However, the argument used to show these facts in Exercise 6.9 is the same for] to be DES reduced to 8 rounds just 8 rounds. Thus, denoting DES (including swapping the two halves before outputting the result),] k (DES] k (x)) = x. Equivalently, it follows that for every x: DES −1] k (x) = DES] k (x). Now, the it follows that for every x: DES] k (DES] k (x)) and DESk (x) is that only difference between DES]] in DES k (DES k (x)) the halves are swapped after 8 rounds. If an input x is such that (L8 , R8) fulfills that L8 = R8 , then this implies] k (DES] k (x)) = DESk (x). that for such an x we have that DES]] k (x)) = x. Thus, However, we have already seen that DES k (DES we concludes that DESk (x) = x (for k a weak key and for x with the property that L8 = R8). (b) In order to find such an x, take any arbitrary 32-bit value z and set L8 = R8 = z. Then, for a weak key k, compute x = (L0 , R0) =] k (L8 , R8). As we have already shown, it also holds that DES −1] k (L8 , R8). Thus, DES] k (DES] k (x)) = DESk (x) = x = DES DESk (x) = x. 6.13 This question illustrates an attack on two-key triple encryption. Let F be a block cipher with n-bit block length and key length, and set def Fk′ 1 ,k2 (x) = Fk1 (Fk−1 (Fk1 (x))). 2 (a) Assume that given a pair (m1 , m2) it is possible to find in constant time all keys k2 such that m2 = Fk−1 (m1). Show how to recover 2 the entire key for F ′ (with high probability) in time roughly 2n using three known input/output pairs. (b) In general, it will not be possible to find k2 as above in constant time. However, show that by using a preprocessing step taking 2n Practical Constructions of Symmetric-Key Primitives 65 time it is possible, given m2 , to find in (essentially) constant time all keys k2 such that m2 = Fk−1 (0n). 2 (c) Assume k1 is known and that the pre-processing step above has already been run. Show how to use the value y = Fk′ 1 ,k2 (x) for a single chosen plaintext x to determine k2 in constant time. (d) Put the above components together to devise an attack that recovers the entire key of F ′ by running in roughly 2n time and requesting the encryption of roughly 2n chosen inputs. Solution: (a) Let (x1 , y1), (x2 , y2), (x3 , y3) be three plaintext/ciphertext pairs that were computed using Fk′ 1 ,k2 (for unknown keys k1 , k2). Initialize a set K = ∅. Then for every key kˆ1 ∈ {0, 1}n do: i. Compute m1 = Fkˆ1 (x1) and m2 = Fkˆ−1 (y1). 1 ii. Find all keys kˆ2 such that m2 = Fkˆ−1 (m1). Call this set S. For 2 each kˆ2 ∈ S do: ? ? If y2 = Fkˆ′ ,kˆ (x2) and y3 = Fkˆ′ ,kˆ (x3), then add (kˆ1 , kˆ2) 1 2 1 2 to K. Let us first analyze the running time of the attack. For a given kˆ1 , the running time of the loop above is linear in |S| (under the assumption that we can find all keys kˆ2 such that m2 = Fkˆ−1 (m1) 2 in constant time). Furthermore, for a given (m1 , m2) we expect S to contain only a single key kˆ2 . Since the algorithm tries all possible values for kˆ1 , the total running time is roughly 2n . It is easy to see that the correct key (k1 , k2) will be in K. We argue that with high probability this is the only key in K. Let Si be the set S in the ith iteration of the algorithm. As we have said above, we expect that each set Si contains only a single key kˆ2 . Summing over all iterations of the algorithm, we expect a total of def 2n keys in S∗ = ∪i Si . But the probability that an incorrect key ∗ in S gets put into K is roughly 2−2n (because this is roughly the probability that an incorrect key (kˆ1 , kˆ2) satisfies y2 = Fkˆ′ ,kˆ (x2) 1 2 and y3 = Fkˆ′ ,kˆ (x3)). Taking a union bound over all incorrect 1 2 keys in S ∗ , we expect an incorrect key to get put into K only with probability 2−n . (b) Using 2n preprocessing time, we can compute all pairs (m2 , k2) such that m2 = Fk−1 (0n) and put these in a list, sorted by m2 . 2 Then given any m2 , we can find in essentially constant time all keys k2 such that (m2 , k2) is in the list. (Note that, in expectation [and treating F as a random function], for any given m2 there is only one k2 such that m2 = Fk−1 (0n).) 2 66 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) If k1 is already known, then we can compute x = Fk−1 (0n) and ask 1 for an encryption of x; let the result be y. Then, given y we can compute m2 = Fk−1 (y). By the definition of F ′ we have that m2 = 1 −1 n Fk2 (0). This is because y = Fk′ 1 ,k2 (x) = Fk1 (Fk−1 (Fk1 (x))) = 2 −1 n −1 −1 n Fk1 (Fk2 (0))) and so m2 = Fk1 (y) = Fk2 (0). We can therefore use the method of the previous step to find k2 in constant time. (d) The full attack works as follows. First run the preprocessing step as described in part (b). Then for every k1 ∈ {0, 1}n , run the attack described in part (c). This will result in a set of 2n possible values for the key (one for each value of k1). Using the plaintext/ciphertext pairs we already have, we can narrow this down to the correct key. For each k1 ∈ {0, 1}n we request the encryption of only one chosen plaintext, for a total of 2n chosen plaintexts. Each step of the attack takes time 2n . We remark that 2n memory is also required. 6.14 Say the key schedule of DES is modified as follows: the left half of the master key is used to derive all the sub-keys in rounds 1–8, while the right half of the master

key is used to derive all the sub-keys in rounds 9-16. Show an attack on this modified scheme that recovers the entire key in time roughly 228 . Solution: The solution to this exercise is to run a meet-in-the-middle attack on DES as follows. Use an encryption oracle to obtain plaintext/ciphertext pairs (x1 , y1), (x2 , y2), (x3 , y3). Then, prepare a list of values (z1 , k1) where k1 ∈ {0, 1}28 and z1 is the result of 8 rounds of DES encryption of x1 using k1 . Likewise, prepare another list of values (z2 , k2) where k2 ∈ {0, 1}28 and z2 is the result of 8 rounds of DES decryption of y1 using k2 . As in the meet-in-the-middle attack on 2DES, it is possible to find pairs k1 , k2 that constitute candidate keys and then verify them using (x2 , y2) and (x3 , y3). 6.15 Let f : {0, 1}m × {0, 1}ℓ → {0, 1}ℓ and g : {0, 1}n × {0, 1}ℓ → {0, 1}ℓ be block ciphers with m > n, and define Fk1 ,k2 (x) = fk1 (gk2 (x)). Show a key-recovery attack on F using time O(2m) and space O(ℓ · 2n). Solution: This is a generalization of the meet-in-the-middle attack. Assume f = O(n). Briefly, given a plaintext/ciphertext pair (x, y), construct a list of size 2n of all the pairs (k2 , z = gk2 (x)), and sort this list by the second element. Next, for every k1 ∈ {0, 1}m , search for fk1−1 (y) in the list. If it exists, then (k1 , k2) is a candidate for the key. By using a few plaintext/ciphertext pairs (with the exact number depending on f), a single candidate key remains. The time to carry out the attack is dominated by the O(2m) Practical Constructions of Symmetric-Key Primitives 67 time to enumerate over all k1 (technically, if binary search is used, this requires time O(m · 2m)). 6.16 Define DESYk,k′ (x) = DESk (x ⊕ k′). The key length of DESY is 120 bits. Show a key-recovery attack on DESY taking time and space ≈ 264 . Solution: The observation is that an optimized meet-in-the-middle attack can be used here. Let {(xi , yi)} be a set of plaintext/ciphertext pairs. For every k ∈ {0, 1}56 compute z1 = DESk−1 (y1) and set k′ = x1 ⊕ z1 . That is, for every guess of k, we obtain a single possibility for k′ . The other plaintext/ciphertext pairs can then be used to check whether this candidate key (k, k′) is correct. This attack requires time 256 and constant space. 6.17 Choose random S-boxes and mixing permutations for SPNs of different sizes, and develop differential attacks against them. We recommend trying five-round SPNs with 16-bit and 24-bit block lengths, using Sboxes with 4-bit input/output. Write code to compute the differential tables, and to carry out the attack. No solution given. 6.18 Implement the time/space tradeoff for 40-bit DES (i.e., fix the first 16 bits of the key of DES to 0). Calculate the time and memory needed, and empirically estimate the probability of success. Experimentally verify the increase in success probability as the number of tables is increased. (Warning: this is a big project!) No solution given. 6.19 For each of the following constructions of a compression function h from a block cipher F , either show an attack or prove collision resistance in the ideal-cipher model: (a) h(k, x) = Fk (x). (b) h(k, x) = Fk (x) ⊕ k ⊕ x. (c) h(k, x) = Fk (x) ⊕ k. Solution: In each case we try the proof approach used to prove Theorem 6.5. (a) There are two ways the attacker can learn the value of h on some input: If it queries Fk (x) and gets response y, then it learns that h(k, x) = y; if it queries Fk−1 (y) and gets response x, then it learns that h(k, x) = y. Observe that here the attacker has complete control over the output of h on some input. Therefore, it is easy to find a collision. In fact, this suggests an attack: 68 Introduction to Modern Cryptography – 2nd Edition Solutions Manual pick arbitrary y and distinct k, k′ . Compute x := Fk−1 (y) and x′ := Fk′−1 (y) and output (k, x) and (k′ , x′). (b) ... (c) There are two ways the attacker can learn the value of h on some input. If it queries Fk (x) and gets response y, then it learns that h(k, x) = y ⊕ k. ... pick (distinct) k, k′ , y, y′ such that k ⊕ y = k′ ⊕ y′ , then compute x := Fk−1 (y) and x′ := Fk′−1 (y′). Then h(k, x) = h(k′ , x′). 6.20 Consider using DES to construct a compression function in the following way: Define h : {0, 1}112 → {0, 1}64 as h(x1 , x2) = DESx1 (DESx2 (064)) where |x1 | = |x2 | = 56. (a) Write down an explicit collision in h. (b) Show how to find a preimage of an arbitrary value y (that is, x1 , x2 such that h(x1 , x2) = y) in roughly 256 time. (c) Show a more clever preimage attack that runs in roughly 232 time and succeeds with high probability. Solution: ... Practical Constructions of Symmetric-Key Primitives 69 (c) The basic idea is that we can run a modified version of the meet-in-the-middle attack from part (b), using the fact that there is no "right" answer. (That is, we are not searching for some fixed key x1 , x2 such that DESx1 (DESx2 (064)) = y; rather, we are searching for any key satisfying this constraint.) We modify the attack as follows: choose 228 random values of x2 and for each such value compute z := DESx2 (064) and store (z, x2) in a list L. Then choose 228 values of x1 and for each such value compute z := DESx1−1 (y) and store (z, x1) in a list. Roughly speaking, we can view each z-value as being "random", and therefore expect a "match" to occur with constant probability. Such a match immediately yields a pre-image for the given value y. 6.21 Let F be a block cipher for which it is easy to find fixed points for some keys: namely, there is a set K of keys for which it is easy to find inputs x for which Fk (x) = x. Find a collision in the Davies-Meyer construction when applied to F. (Consider this in light of Exercise 6.12.) Solution: Davies-Meyer is defined by h(k, x) = Fk (x) ⊕ x. ... Chapter 7 Theoretical Constructions of Symmetric-Key Primitives – Solutions 7.1 Prove that if there exists a one-way function, then there exists a one-way function f such that f (0n) = 0n for every n. Note that for infinitely many values y, it is easy to compute f −1 (y). Why does this not contradict one-wayness? Solution: We provide a painfully detailed proof. Let f be a one-way function (that exists by the assumption) and define g(x) = f (x) for every x ≠ 0|x| and g(0n) = 0n for every n. Clearly, g fulfills the requirements. It remains to prove that it is one-way. First, g is efficiently computable. ... Theoretical Constructions of Symmetric-Key Primitives ... 7.2 ... 7.3 ... 7.4 ... 7.5 ... 7.6 ... 7.7 ... 7.8 ... 7.9 ... 7.10 ... 7.11 ... 7.12 ... 7.13 ... 7.14 ... 7.15 ... 7.16 ...

Output 1 ? iff L0 ⊕ L′0 = L2 ⊕ L′2 . Observe that if O is a permutation chosen at random, then (L2 , R2) and (L′2 , R2′) are uniform subject to the constraint that they are unequal; Theoretical Constructions of Symmetric-Key Primitives 83 thus, the probability that A outputs 1 is roughly 2−n . On the other hand, if O is a 2-round Feistel network with round functions f1 , f2 then L2 = R1 = L0 ⊕ f1 (R0) and, similarly, L′2 = L′0 ⊕ f1 (R0). So, L2 ⊕ L′2 = L0 ⊕ f1 (R0) ⊕ L′0 ⊕ f1 (R0) = L0 ⊕ L′0 , and so A outputs 1 with probability 1 in this case. A thus has a significant advantage in distinguishing between a two-round Feistel network and a truly random permutation. 7.17 Prove that a three-round Feistel network using pseudorandom round functions (as in Equation (7.16)) is not strongly pseudorandom. Solution: We show an attack that works (with high probability) regardless of what round functions are used. The attack exploits the fact that we can introduce a known "shift" δ in the input to the second round-function of the Feistel network in two ways: by XORing δ into the left half of the input when querying the function in the forward direction (and keeping the right half fixed), or by XORing δ into the right half of the input when querying the function in the reverse direction (and keeping the left half fixed). It is easiest to see this by tracing through the effects of these modifications in Figure 7.3. We now give the details. The attacker A, given access to an oracle O : {0, 1}2n → {0, 1}2n and its inverse O−1 : {0, 1}2n → {0, 1}2n works as follows: choose arbitrary L0 , R0 , δ ∈ {0, 1}n with δ ≠ 0n . Then: (a) Query O(L0 , R0) to obtain (L3 , R3). (b) Query O−1 (L3 , R3 ⊕ δ) to obtain (L′0 , R0′). (c) Query O(L0 ⊕ δ, R0) to obtain (L′′3 , R3′′). ? Output 1 iff L3 ⊕ R0′ = L′′3 ⊕ R0 . When O is a random permutation, A outputs 1 with only negligible probability. To see this, note that (L3 , R3 ⊕ δ) ≠ (L3 , R3) and so (L′0 , R0′) is uniform subject to being unequal to (L0 , R0) (because O is a permutation). That (L′0 , R0′) = (L0 ⊕δ, R0) is, therefore, roughly 2−2n . Assuming (L′0 , R0′) ≠ (L0 ⊕ δ, R0), the value (L′′3 , R3′′) is uniform subject to being different from both (L3 , R3) and (L3 , R3 ⊕ δ), and so L3 ⊕ R0′ = L′′3 ⊕ R0 only with probability roughly 2−n . 84 Introduction to Modern Cryptography – 2nd Edition Solutions Manual On the other hand, if O is a 3-round Feistel network with round functions f1 , f2 , f3 then ( ) L3 ⊕ R0′ = L3 ⊕ L3 ⊕ f2 (R3 ⊕ δ ⊕ f3 (L3)) = f2 (R3 ⊕ δ ⊕ f3 (L3)) and (7.2) ( ) L′′3 ⊕ R0 = R0 ⊕ f2 (L0 ⊕ δ ⊕ f1 (R0)) ⊕ R0 = f2 (L0 ⊕ δ ⊕ f1 (R0)). (7.3) Since R3 ⊕ f3 (L3) = L0 ⊕ f1 (R0), we see that Equations (7.2) and (7.3) are equal, and so A outputs 1 with probability 1 in this case. A thus has a significant advantage in distinguishing between a three-round Feistel network and a truly random permutation. 7.18 Consider the keyed permutation F ∗ defined by Fk∗ (x) = FeistelFk ,Fk ,Fk (x). def (Note that the same key is used in each round.) Show that F ∗ is not pseudorandom. Solution: When the same key is used in all rounds, computing F ∗ in the forward direction is essentially the same as inverting it. In particular, for every L0 , R0 ∈ {0, 1}n/2 it holds that Fk∗ (L0 , R0) = F ∗ −1 k (R0 , L0). In order to see this, observe that the only difference between computing F ∗ in the forward or reverse direction is that in the forward direction the first input to Fk in the first round is R0 , whereas in the reverse direction the first input to Fk is R2 = L3 (see Figure 7.3 in the book). This observation yields the following distinguisher D: set x = 0n and query the oracle, receiving back y. Denote y = (L3 , R3). Then, query x′ = (R3 , L3) to the oracle. If the response equals 0n then output 0; otherwise, output 1. Based on what we have explained above [∗] Pr DFk (·) (1n) = 1 = 1 whereas [ ] Pr Df (·) (1n) = 1 = 2−n since for a pseudorandom permutation, the probability that f (f (0n)) = 0n is exactly 2−n . (In order to see this, choose f by first choosing the output of 0n , and then continuing to choose the outputs one at a time.) 7.19 Let G be a pseudorandom generator with expansion factor ℓ(n) = n + 1. Prove that G is a one-way function. Theoretical Constructions of Symmetric-Key Primitives 85 Solution: Assume by contradiction that there is a probabilistic polynomialtime algorithm A and a non-negligible function ε(·) such that Pr[InvertA,G (n) = 1] = ε(n). We construct a distinguisher D for the pseudorandom generator as follows. Upon input r of length n + 1, distinguisher D invokes A upon 1n and r; let s be A's output. If r = G(s) then D outputs 1; otherwise, D outputs a random bit b ← {0, 1}. We have: Prs∈{0,1}n [D(G(s)) = 1] 1 · Pr[InvertA,G (n) = 1] 2 1 1 ε(n) = ε(n) + · (1 − ε(n)) = + . 2 2 2 1 · Pr[InvertA,G (n) = 1] + In contrast, 1 . 2 This holds because the range of G (on inputs of length n) is of size at most 2n , and so at least half the strings in {0, 1}n+1 are not even in the range of G (and so there doesn't exist an s ∈ {0, 1}n such that G(s) = r). We conclude that Prr∈{0,1}n+1 [D(r) = 1] ≤ Prs∈{0,1}n [D(G(s)) = 1] − Prr∈{0,1}n+1 [D(r) = 1] ≥ ε(n) 2 in contradiction to the assumption that G is a pseudorandom generator. c c 7.20 Let X , Y, Z be probability ensembles. Prove that if X ≡ Y and Y ≡ Z, then X ≡ Z. Solution: Let D be a probabilistic polynomial-time algorithm. We wish to show that there exists a negligible function negl(n) such that: Pr [D(1n , x)) = 1] − Pr [D(1n , z) = 1] ≤ negl(n). (7.4) x←Xn z←Zn By the assumption, there exist negligible functions negl1 (n) and negl2 (n) such that: Pr [D(1n , x)) = 1] − Pr [D(1n , y) = 1] ≤ negl1 (n) x←Xn y←Yn Pr [D(1n , y)) = 1] − Pr [D(1n , z) = 1] ≤ negl2 (n). y←Yn z←Zn 86 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Thus, Pr [D(1n , x)) = 1] − Pr [D(1n , z) = 1] x←Xn z←Zn = Pr [D(1n , x)) = 1] − Pr [D(1n , y)) = 1] x←Xn y←Yn + Pr [D(1n , y)) = 1] − Pr [D(1 , z) = 1] y←Yn z←Zn n n ≤ Pr [D(1 , x)) = 1] − Pr [D(1 , y) = 1] x←Xn y←Yn n n + Pr [D(1 , y)) = 1] − Pr [D(1 , z) = 1] n y←Yn n z←Zn ≤ negl1 (n) + negl2 (n), where the first inequality is by the triangle inequality. Since the sum of negligible functions is negligible, we conclude that Equation (7.4) holds. 7.21 Prove Theorem 7.32. Solution: The proof is by reduction. We show that if there exists a probabilistic polynomial-time distinguisher D that distinguishes X from Y with non-negligible success, then there exists a probabilistic polynomial-time distinguisher D′ that distinguishes a single sample of Xn from a single sample of Yn with non-negligible success. Our proof uses a hybrid argument. Let D be a probabilistic polynomial-time distinguisher and let ε(n) be a function such that [ ] [ ] Pr D(Xn(1) , . . . , Xn(p(n))) = 1 − Pr D(Yn(1) , . . . , Yn(p(n))) = 1 = ε(n). (7.5) For every i, we define a hybrid random variable Hni as a sequence containing i independent copies of Xn followed by p(n) − i independent copies of Yn . That is: ( ) Hni = Xn(1) , . . . , Xn(i) , Yn(i+1) , . . . , Yn(p(n)) (1) (p(n)) Denote X n = (Xn , . . . , Xn) and likewise for Y n . Then, we have p(n) that Hn0 = Y n and Hn = X n . Thus, Pr[D(X n) = 1] − Pr[D(Y n) = 1] = Pr[D(Hnp(n)) = 1] − Pr[D(Hn0) = 1] p(n)−1 p(n)−1 ∑ ∑ i i+1 = Pr[D(Hn) = 1] − Pr[D(Hn) = 1] i=0 i=0 Theoretical Constructions of Symmetric-Key Primitives 87 where the second equality follows from the fact that the only remaining p(n) terms in this telescopic sum are Pr[D(Hn0) = 1] and Pr[D(Hn) = 1]. We now construct a probabilistic polynomial-time distinguisher D′ for a single sample of Xn and Yn . Upon input some sample α of Xn or Yn , D′ chooses a random i ← {0, . . . , p(n) − 1}, generates the vector (1) (i) (i+2) (p(n)) H n = (Xn , . . . , Xn , α, Yn , . . . , Yn), invokes D on the vector H n , and outputs whatever D does. Now, if α is distributed according to Xn , then H n is distributed exactly like Hni+1 . In contrast, if α is distributed according to Yn , then H n is distributed exactly like Hni . (Note that we use the independence of the samples in making this argument.) Furthermore, each i is chosen with probability exactly 1/p(n). Therefore, Pr[D′ (Xn) = 1] = and Pr[D′ (Yn) = 1] = p(n)−1 ∑ 1 · Pr[D(Hni+1) = 1] p(n) i=0 p(n)−1 ∑ 1 · Pr[D(Hni) = 1] p(n) i=0 It therefore follows that: |Pr[D′ (Xn) = 1] − Pr[D′ (Yn) = 1]| p(n)−1 p(n)−1 ∑ ∑ 1 i+1 i = · Pr[D(Hn) = 1] − Pr[D(Hn) = 1] p(n) i=0 i=0 1 · Pr[D(Hnp(n)) = 1] − Pr[D(Hn0) = 1] = p(n) ε(n) 1 · Pr[D(X n) = 1] − Pr[D(Y n) = 1] = . = p(n) p(n) By the assumption that X and Y are indistinguishable, we conclude ε(n) that p(n) is negligible, implying that ε(n) itself is negligible. By Equation (7.5), this implies that X is indistinguishable from Y , completing the proof of the theorem. 7.22 Let X = {Xn }n∈N and Y = {Yn }n∈N be computationally indistinguishable probability ensembles. Prove that for any probabilistic polynomialtime algorithm A, the ensembles {A(Xn)}n∈N and {A(Yn)}n∈N are computationally indistinguishable. Solution: Assume there is a probabilistic polynomial-time A, a distinguisher D and a non-negligible function ε(·) such that | Pr[D(1n , A(Xn)) = 1] − Pr[D(1n , A(Yn)) = 1]| = ε(n). 88 Introduction to Modern Cryptography – 2nd Edition Solutions Manual We construct a distinguisher D′ that receives a sample z (from Xn or Yn), runs A on the input (i.e. z, but not the 1n part) and then runs D on 1n together with the result from A. Finally, D′ outputs whatever D does. Clearly, for every z, D′ (1n , z) = 1 if and only if D(1n , A(z)) = 1. Therefore, | Pr[D′ (1n , Xn) = 1] − Pr[D′ (1n , Yn) = 1]| = ε(n) a contradiction. Chapter 8 Number Theory and Cryptographic Hardness Assumptions – Solutions 8.1 Let G be an abelian group. Prove that there is a unique identity in G, and that every element g ∈ G has a unique inverse. Solution: Let G be a group, and let e, e′ ∈ G with eg = ge = g and e′ g = ge′ = g for all g ∈ G. Then e′ = e · e′ = e and so e and e′ are the same element. Let 1 denote the identity in G. Fix g ∈ G and let h, h′ ∈ G be such that gh = hg = 1 and gh′ = h′ g = 1. Then hgh′ = (hg)h′ = 1 · h′ = h′ and hgh′ = h(gh′) = h · 1 = h, and so h′ = h. 8.2 Show that Proposition 8.36 does not necessarily hold when G is infinite. Solution: Take G = R+ , i.e., the group of all positive real numbers with respect to multiplication. Let H = {1} ∪ {2, 4, 6, 8, . . .} be a subset of G. Note that H satisfies the conditions of Proposition 8.36, but is not a subgroup since no elements in H (other than 1) have inverses in H. 8.3 Let G be a finite group, and g ∈ G. Show that ⟨g⟩ is a subgroup of G. Is the set {g 0 , g 1 , . . .} necessarily a subgroup of G when G is infinite? Solution: We verify that ⟨g⟩ satisfies the conditions of Proposition 8.36. Since g 0 = 1, clearly ⟨g⟩ is nonempty. Furthermore, for any a, b ∈ ⟨g⟩ we know that there exist positive x, y such that a = g x and b = g y . But then ab = g x+y ∈ ⟨g⟩. So ⟨g⟩ is a subgroup when G is finite. The set {g 0 , g 1 , . . .} need not be a subgroup of G when G is infinite. For example, consider the group R+ of all positive real numbers with respect def to multiplication. The set S = {20 , 21 , 22 , . . .} is not a subgroup of R+ since, for example, the element 2 ∈ S has no inverse in S. 8.4 This question concerns the Euler phi function. 89 90 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (a) Let p be a prime and e ≥ 1 an integer. Show that ϕ(pe) = pe−1 (p − 1). (b) Let p, q be relatively prime. Show that ϕ(pq) = ϕ(p) · ϕ(q). (You may use the Chinese remainder theorem.) (c) Prove Theorem 8.19. Solution: (a) The integers in {1, . . . , pe } that are not relatively prime to pe are exactly the multiples of p, and there are exactly pe /p = pe−1 such integers. So ϕ(pe) = pe − pe−1 = pe−1 (p − 1). (b) We provide a solution that does not use the Chinese remainder theorem. Let p and q be relatively prime. Consider the integers {1, . . . , pq} arranged in an array as follows: 1 p+1 2 p+2 .. . .. . p 2p · · · (q − 1)p + 1 · · · (q − 1)p + 2 .. . .. . ··· qp For any r having a factor in common with p, every element in the row r p + r 2p + r · · · (q − 1)p + r has a factor in common with p and hence also has a factor in common with pq. Eliminate those rows from consideration, and consider the remaining ϕ(p) rows. Let s p + s 2p + s · · · (q − 1)p + s be such a row (i.e., gcd(s, p) = 1). We claim that [s mod q] [p + s mod q] [2p + s mod q] · · · [(q − 1)p + s mod q] contains the elements of Zq in permuted order. This follows since there are q such elements, and ip + s = jp + s mod q implies j = i (using gcd(p, q) = 1). From this it follows that each remaining row contains exactly ϕ(q) elements relatively prime to q. The leaves a def total of ϕ(pq) = ϕ(p) · ϕ(q) elements relatively prime to pq. ∏ (c) Let N = i pei i . Parts (a) and (b) immediately give ∏ ∏ ϕ(N) = ϕ(pei 1) = piei −1 (pi − 1). i i Number Theory and Cryptographic Hardness Assumptions 91 8.5 Compute the final two (decimal) digits of 31000 (by hand). Solution: Note that |Z∗100 | = ϕ(100) = 40. So 31000 = 31000 mod 40 = 30 = 1 mod 100. 8.6 Compute [1014,800,000,002 mod 35] (by hand). Solution: We use the Chinese remainder theorem. First, 1014,800,000,002 = 14,800,000,002 = 1 mod 5. Next, 1014,800,000,002 = 34,800,000,002 mod 6 = 32 = 2 mod 7. Finally, we can explicitly compute (by trial-and-error) that (1, 2) ↔ 16. So, the answer is 16. 8.7 Compute [4651 mod 55] (by hand) using the Chinese remainder theorem. Solution: 55 = 11 · 5 and so we will work mod 11 and mod 5. First, [46 mod 11] = 2. Furthermore ϕ(11) = 10 and thus [4651 mod 11] = [251 mod 10 mod 11] = [21 mod 11]. Similarly, [46 mod 5] = 1. We conclude that [4651 mod 55] ↔ (2, 1). We thus need to find the value x ∈ Z55 such that x mod 11 = 2 and x mod 5 = 1. But we have already seen that 46 is such a value. So 46 = 4651 mod 55. 8.8 Prove that if G, H are groups, then G × H is a group. Solution: The conditions of Definition 7.9 are easy to verify. Closure and associativity are obvious. If e, e′ are the identity elements of G and H, respectively, then (e, e′) is the identity of G × H. Finally, any (g, h) ∈ G × H has inverse (g −1 , h−1), where g −1 (resp., h−1) is the inverse of g (resp., h) in G (resp., H). 8.9 Let p, N be integers with p | N . Prove that for any integer X, [[X mod N] mod p] = [X mod p]. Show that, in contrast, [[X mod p] mod N] need not equal [X mod N]. Solution: Let b = [X mod N] so that X can be written as X = aN + b with a ∈ Z. Let β = [b mod p] so that b can be written as b = αp + β with α ∈ Z. Then X = aN + αp + β and, since p | N , we therefore have [X mod p] = β = [[X mod N] mod p], as desired. On the other hand, let X = 15, N = 10, and p = 5. Then we have [X mod N] = 5 but [[X mod p] mod N] = 0. 92 Introduction to Modern Cryptography – 2nd Edition Solutions Manual 8.10 Corollary 8.21 shows that if N = pq and ed = 1 mod ϕ(N) then for d all x ∈ Z∗N we have (xe) = x mod N . Show that this holds for all x ∈ {0, . . . , N − 1}. Solution: If ed = 1 mod ϕ(N) then, because ϕ(N) = (p − 1)(q − 1) = ϕ(p) · ϕ(q), it also holds that ed = 1 mod ϕ(p). So for any x ∈ Z∗p d we know that (xe) = x mod p. But for x = 0 it also holds, trivially, d that (xe) = x mod p. We conclude that for any x ∈ Zp , it holds that d (xe) = x mod p. The same holds with p replaced by q. Now let x ∈ ZN . Then x ↔ (xp , xq) with xp ∈ Zp and xq ∈ Zq . Now d d (xe) ↔ ((xp , xq)e) ([( )] [( )]) d d = xep mod p , xeq mod q = (xp , xq) ↔ x. 8.11 Complete the details of the proof of the Chinese remainder theorem, showing that Z∗N is isomorphic to Z∗p × Z∗q . Solution: Let f be as defined in Theorem 8.24, restricted to domain Z∗N . It follows from the proof in the text that f is a bijection. It therefore only remains to show that for any a, b ∈ Z∗N we have f (a ·N b) = f (a) f (b), where ·N denotes multiplication modulo N and denotes the group operation in Z∗p × Z∗q (namely, multiplication modulo p in the first component and modulo q in the second component). To see that this is true, note that f (a ·N b) = ([(a ·N b) mod p], [(a ·N b) mod q]) = ([(a · b) mod p], [(a · b) mod q]) = ([a mod p], [a mod q]) ([b mod p], [b mod q]) = f (a) f (b). 8.12 This exercise develops an efficient algorithm for testing whether an integer is a perfect power. ˆ e for some integers N ˆ , e > 1 then e ≤ ∥N ∥ + 1. (a) Show that if N = N (b) Given N and e with 2 ≤ e ≤ ∥N ∥ + 1, show how to determine in ˆ with N ˆe = N. poly(∥N ∥) time whether there exists an integer N (c) Given N , show how to test in poly(∥N ∥) time whether N is a perfect power. Solution: ˆ e then e = log ˆ N ≤ log2 N ≤ ∥N ∥ + 1. (a) If N = N N Number Theory and Cryptographic Hardness Assumptions 93 ALGORITHM 8.1-S Computing N 1/e Input: N, e > 1 Output: N 1/e if this is an integer; false otherwise low := 1, high := N while low ≤ high: { ˆ := ⌊(low + high)/2⌋ N ˆ e = N then return N ˆ and halt if N ˆ e > N then high := N ˆ −1 if N e ˆ ˆ if N < N then low := N + 1 } return false (b) Consider Algorithm 8.1-S. Clearly, the algorithm always outputs false when N 1/e is not an integer. Otherwise, the algorithm outputs N 1/e in at most O(log N) = O(∥N ∥) iterations. (c) Run Algorithm 8.1-S for each of at most O(∥N ∥) values of e. 8.13 Given N and a ∈ Z∗N , show how to test in polynomial time whether a is a strong witness that N is composite. Solution: See Algorithm 8.2-S. ALGORITHM 8.2-S Testing strong witnesses Input: odd integer N , a ∈ Z∗N Output: Decide whether a is a strong witness Compute r, u such that N − 1 = 2r u with u odd x := [au mod N] if x = ±1 then return false for i = 1 to r − 1: { x := [x2 mod N] if x = N − 1 then return false } return true 8.14 Fix N, e with gcd(e, ϕ(N)) = 1, and assume there is an adversary A running in time t for which Pr [A ([xe mod N]) = x] = 0.01, where the probability is taken over uniform choice of x ∈ Z∗N . Show that it is possible to construct an adversary A′ for which Pr [A′ ([xe mod N]) = x] = 0.99 for all x. The running time t′ of A′ should be polynomial in t and ∥N ∥. 94 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Solution: Let s be a parameter, fixed later. Construct A′ as follows: Algorithm A′ On input N, y, e do: (a) For i = 1 to s: i. Choose ri ← Z∗N . ii. Run A([(ri)e · y mod N]) to obtain xi . iii. If (xi)e = (ri)e · y mod N then output [xi /ri mod N] and terminate. (b) If the algorithm has not yet terminated, output fail. Let y be arbitrary. The key point to note is that, in every iteration, A is run on a uniform element of Z∗N , irrespective of how y is distributed. This is so since ri is uniform, hence [(ri)e mod N] is uniform (since raising to eth powers is a permutation), and thus [(ri)e · y mod N] is uniform (because Z∗N is a group). Furthermore, if A ever correctly computes an eth root in any iteration, then A′ outputs the eth root of y: this follows since (xi)e = (ri)e · y mod N implies (xi /ri)e = y mod N . Combining the observations above and setting s = 100 ln 100, we see that the probability that A′ fails to output an inverse is ( )100 ln 100 (( )100)ln 100 1 1 1 1− = 1− ≤ e− ln 100 = 100 100 100 (using Proposition A.2). The running time of A′ is O(t · poly(∥N ∥)). 8.15 Formally define the CDH assumption. Prove that hardness of the CDH problem relative to G implies hardness of the discrete-logarithm problem relative to G, and that hardness of the DDH problem relative to G implies hardness of the CDH problem relative to G. Solution: We first formally define what it means for the CDH problem to be hard. The CDH experiment CDHA,G (n): (a) Run G(1n) to obtain (G, q, g), where G is a cyclic group of order q (with ∥q∥ = n), and g is a generator of G. (b) Choose h1 , h2 ← G. (c) A is given G, q, g, h1 , h2 , and outputs h3 ∈ G. (d) The output of the experiment is defined to be 1 if h3 = DHg (h1 , h2), and 0 otherwise. As described, it is unclear how to test whether h3 = DHg (h1 , h2) efficiently. However, by choosing uniform h1 with logg h1 known (which Number Theory and Cryptographic Hardness Assumptions 95 can be done by choosing uniform x1 ∈ Zq and setting h1 := g x1) it becomes easy to perform. DEFINITION We say that the CDH problem is hard relative to G if for all probabilistic polynomial-time algorithms A there exists a negligible function negl such that Pr[CDHA,G (n) = 1] ≤ negl(n). Assume the CDH problem is hard relative to G. Let A be a probabilistic def polynomial-time algorithm and set ε(n) = Pr[DLogA,G (n) = 1]. We show how A can be used by a probabilistic polynomial-time algorithm A′ to solve the CDH problem with success probability ε(n): Algorithm A′ : The algorithm is given G, q, g, h1 , h2 as input. (a) Run A(G, q, g, h1) and obtain output x1 . (b) Return hx2 1 . Observe that the input given to A when run as a subroutine by A′ is distributed exactly as in experiment DLogA,G (n). So with probability ε(n) it holds that x1 = logg h1 . Whenever this occurs, A′ outputs the correct answer DHg (h1 , h2). Since the CDH problem was assumed to be hard, this implies that ε(n) is negligible and so the discrete logarithm problem must be hard as well. We now proceed to prove that the hardness of the DDH problem relative to G implies the hardness of the CDH problem relative to G. def Let A be a probabilistic polynomial-time algorithm and set ε(n) = Pr[CDHA,G (n) = 1]. We show how A can be used by a probabilistic polynomial-time algorithm A′ to solve the DDH problem with roughly the same probability: Algorithm A′ : The algorithm is given G, q, g, h1 , h2 , h3 as input. (a) Run A(G, q, g, h1 , h2) and obtain output h′3 . (b) If h′3 = h3 output 1; else output 0. We need to analyze the probability that A′ outputs 1 when the final component of its input h3 is equal to DHg (h1 , h2), as compared to the probability that it outputs 1 when h3 is chosen uniformly at random from G. The input given to A when run as a subroutine by A′ is distributed exactly as in experiment CDHA,G (n). So with probability ε(n), it holds that h′3 = DHg (h1 , h2). It follows that when h3 = DHg (h1 , h2), algorithm A′ outputs 1 with probability at least ε(n). 96 Introduction to Modern Cryptography – 2nd Edition Solutions Manual On the other hand, when h3 is chosen uniformly at random from G, then regardless of the behavior of A′ the probability that h′3 = h3 is exactly 1/|G| = 1/q. Thus, Pr[A(G, q, g, g x , g y , g z) = 1] − Pr[A(G, q, g, g x , g y , g xy) = 1] ≥ ε(n) − 1/q. Since the above must be negligible by the assumption that the DDH problem is hard, and 1/q is negligible, it follows that ε(n) must be negligible. We conclude that the CDH problem is hard as well. 8.16 Determine the points on the elliptic curve E : y 2 = x3 + 2x + 1 over Z11 . How many points are on this curve? Solution: This simply involves finding all values of x for which x3 +2x+ 1 is either 0 or a quadratic residue modulo 11. The quadratic residues modulo 11 are 1, 3, 4, 5, and 9. So we have the points (0, 1), (0, 10), (1, 2), (1, 9), (3, 1), (3, 10), (5, 2), (5, 9), (6, 3), (6, 8), (8, 1), (8, 10), (9, 0), (10, 3), (10, 8), plus the point at infinity. 8.17 Consider the elliptic-curve group from Example 8.67. (See also Example 8.69.) Compute (1, 0)+(4, 3) in this group by first converting to projective coordinates and then using Equations (8.3) and (8.4). Solution: In projective coordinates these points can be represented as (1, 0, 1), (4, 3, 1), and (4, 3, 1). To compute (1, 0, 1) + (4, 3, 1) we first compute u = 3, v = 3, and w = 6. Then (1, 0, 1) + (4, 3, 1) = (4, 2, 6) = (−4, −2, −6) = (3, 5, 1). To compute (3, 5, 1)+(4, 3, 1) we proceed similarly. The answer, in affine coordinates, is (4, 4) in agreement with Exercise 8.69. 8.18 Prove the fourth statement in Proposition 8.68. Solution: To compute the slope of the line tangent to E at P1 = P2 = (x1 , y1), we first compute the partial derivatives of E as 2y · dy = 3x2 · dx + A · dx, so that the slope at (x1 , y1) is given by def m = [2] dy 3x1 + A (x1 , y1) = mod p . dx 2y1 The rest of the proof follows the derivation of Proposition 8.68 verbatim, using the above as the value of m. Number Theory and Cryptographic Hardness Assumptions 97 8.19 Can the following problem be solved in polynomial time? Given a prime p, a value x ∈ Z∗p−1 , and y := [g x mod p] (where g is a uniform value in Z∗p), find g, i.e., compute y 1/x mod p. If your answer is "yes," give a polynomial-time algorithm. If your answer is "no," show a reduction to one of the assumptions introduced in this chapter. Solution: The problem can be solved efficiently. Since x ∈ Z∗p−1 (and p − 1 is known), we can compute d := [x−1 mod p − 1]. Then d y d = (g x) = g xd = g [xd mod p−1] = g mod p. 8.20 Let GenRSA be as in Section 8.2.4. Prove that if the RSA problem is hard relative to GenRSA then the construction shown below is a fixedlength collision-resistant hash function. CONSTRUCTION 8.3-S Define (Gen, H) as follows: • Gen: on input 1n , run GenRSA(1n) to obtain N, e, d, and select y ← Z∗N . The key is s := ⟨N, e, y⟩. • H: if s = ⟨N, e, y⟩, then H s maps inputs in {0, 1}3n to outputs in Z∗N . Let f0s (x) = [xe mod N] and f1s (x) = [y · xe mod N] . For a 3n-bit long string x = x1 · · · x3n , define ((( ))) def . H s (x) = fxs3n fxs3n−1 · · · 1 · · · def Solution: We show that an eth root of y can be computed from any collision; given this, a formal proof follows along the lines of the proof of Theorem 8.79. Fix s = ⟨N, e, y⟩, and assume y ≠ 1 (computing an eth root of y is trivial if y = 1). Let x, z be distinct inputs with H s (x) = H s (z). Define x0 = 1, and for i = 1 to 3n, inductively define xi = fxsi (xi−1) (note that x1 , . . . , x3n is the sequence of values produced when computing H s (x)). Define z i analogously. Since x ≠ z, there is some index i with xi ≠ z i (in particular, take i to be the first position where the strings x and z differ, i.e., where xi ≠ zi). But since x3n = z 3n , there must be an index j < 3n with xj ≠ z j but xj+1 = z j+1 . Since f0s , f1s are permutations (this can be easily verified), it must be the case that xj+1 ≠ zj+1 . Assume without loss of generality that xj+1 = 0. The above shows that given a collision we can efficiently 98 Introduction to Modern Cryptography – 2nd Edition Solutions Manual compute distinct elements X, Z ∈ Z∗N with f0s (X) = f1s (Z). Then X e = y · Z e mod N, and we see that X/Z mod N is an eth root of y. 8.21 Consider the following generalization of Construction 8.78: CONSTRUCTION 8.4-S Define a fixed-length hash function (Gen, H) as follows: • Gen: on input 1n , run G(1n) to obtain (G, q, h1) and then select h2 , . . . , ht ← G. Output s := ⟨G, q, (h1 , . . . , ht)⟩ as the key. • H: given a key s = ⟨G, q, (h1 , . . . , ht)⟩ ∏ and input (x1 , . . . , xt) with xi ∈ Zq , output H s (x1 , . . . , xt) := i hxi i . (a) Prove that if the discrete logarithm problem is hard relative to G, and q is prime, then for any t = poly(n) this construction is a fixed-length collision-resistant hash function. (b) Discuss how this construction can be used to obtain compression regardless of the number of bits needed to represent elements of G (as long as it is polynomial in n). Solution: (a) The easiest way to prove this is by reducing to the security of Construction 8.78. That is, we show that any algorithm A finding a collision in Construction 8.4-S can be used as a subroutine to find collisions in Construction 8.78. d H) b = (Gen, b Let Π = (Gen, H) denote Construction 8.4-S and let Π denote Construction 8.78. Given a probabilistic polynomial-time def algorithm A attacking Π, let ε(n) = Pr[Hash-collA,Π (n) = 1]. Conb as follows: struct algorithm Aˆ attacking Π ˆ Algorithm A: The algorithm is given sˆ = ⟨G, q, g, h⟩ as input. i. Set h1 := g and let β1 := 1. Choose i ← {2, . . . , t} and set hi := h. ii. For j ∈ {2, . . . , t} \ {i}, choose βj ← Zq and set hj := g βj . Set s := ⟨G, q, (h1 , . . . , ht)⟩. iii. Run A(s) and obtain outputs (x1 , . . . , xt) and (x′1 , . . . , x′t). iv. If xi ≠ x′i then ∑ A. Set X1 := j≠i βj · xj and X2 := xi . Number Theory and Cryptographic Hardness Assumptions 99 ∑ B. Set X1′ := j≠i βj · x′j and X2′ := x′i . v. Output (X1 , X2) and (X1′ , X2′). We first claim that whenever (x1 , . . . , xt) and (x′1 , . . . , x′t) form a b sˆ. To collision in H s with xi ≠ x′i , then Aˆ outputs a collision in H see this, note that ∏ x H s (x1 , . . . , xt) = j hj j ∏ ( ) xj = hxi i · j≠i g βj ∑ = hxi · g j≠i βj ·xj b sˆ(X1 , X2), = g X1 · hX2 = H b sˆ(X ′ , X ′). Since the values outand similarly H s (x′1 , . . . , x′t) = H 2 1 b sˆ(X1 , X2) = H b sˆ(X ′ , X ′); since xi ≠ x′ we have put by A collide, H 2 1 i b sˆ. X1 ≠ X2′ and so (X1 , X2) and (X1′ , X2′) are a collision in H We now analyze the probability that Aˆ outputs a collision. First note that the distribution on the inputs given to A when run as a subroutine by Aˆ is identical to the distribution in experiment Hash-collA,Π (n), and A therefore outputs a collision (x1 , . . . , xt), (x′1 , . . . , x′t) with probability exactly ε(n). We claim that for at least one value k ∈ {2, . . . , t} we have xk ≠ x′k . Since (x1 , . . . , xt) and (x′1 , . . . , x′t) are distinct, the only other possibility is that 1 ≠ ∏ x x′1 ; in this case, however, it follows from the fact that j hj j = ∏ x′j ′ ′ j hj that (x2 , . . . , xt) and (x2 , . . . , xt) cannot all be the same. (Note that h1 ≠ 1 since it is a generator of G.) ˆ choice of i, since each element A has no information about A's h2 , . . . , ht is a uniform element in G. It follows that xi ≠ x′i with probability at least 1/(t − 1). Combining this with the argument given earlier, we see that Aˆ outputs a collision with probability at b least ε(n)/(t − 1). Since this must be negligible (by security of Π) and t = poly(n), it follows that ε(n) must be negligible as well. (b) The input length of Construction 8.4-S can be t·(n−1) (since ∥q∥ = n), and the output length is given by the length of a bit-string needed to represent an element of G. The latter is p(n) for some polynomial p. Setting t > p(n)/(n − 1), we obtain compression. Chapter 9 Algorithms for Factoring and Computing Discrete Logarithms – Solutions 9.1 In order to speed up the key generation algorithm for RSA, it has been suggested to generate a prime by generating many small random primes, multiplying them together and adding one (of course, then checking that the result is prime). Ignoring the question of the probability that such a value really is prime, what do you think of this method? Solution: This is not a good idea, since if a prime p is generated using this method then p − 1 will have small prime factors and so Pollard's p − 1 algorithm can be applied. def 9.2 In an execution of Algorithm 9.2, define x(i) = F (i) (x(0)). Show that if, in a given execution of the algorithm, there exist i, j ≤ 2n/2 such that x(i) ≠ x(j) but x(i) = x(j) mod p, then that execution of the algorithm outputs p with overwhelming probability. (i) def Solution: Let xp = [x(i) mod p]. As in the proof of Claim 5.10, if (J) (I) xp = xp for 1 ≤ I < J ≤ 2n/2 then there is an i < J such that (2i) (i) xp = xp . In addition (as noted already in Section 9.1.2), x(i) ≠ x(2i) with overwhelming probability if we model F as a random function In iteration i of Algorithm 9.2, x is set equal to x(i) and x′ is set equal to x(2i) . Thus, iteration i of the algorithm will output p with overwhelming probability. 9.3 (a) Show that if ab = c mod N and gcd(b, N) = d, then: i. d | c; ii. a · (b/d) = (c/d) mod (N/d); and iii. gcd(b/d, N/d) = 1. (b) Describe how to use the above to compute discrete logarithms in ZN efficiently even when the base g is not a generator of ZN . Solution: (a) Since ab = c mod N we have ab − αN = c for some integer α. Since d | b and d | N , it follows immediately that d | c, proving (i). 101 102 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Furthermore, a(b/d) − α(N/d) = c/d proving (ii). To prove (iii), let d′ be a positive integer dividing both b/d and N/d. Then dd′ divides both b and N . Since d = gcd(b, N), this implies d′ = 1. (b) Say we are given N along with g, y ∈ ZN , and want to compute x ∈ {0, . . . , N −1} such that x·g = y mod N . If g is a generator then, as discussed in Section 8.2.3, we could compute x := [y · g −1 mod N]. If g is not a generator then gcd(g, N) = d > 1. If d ∤ y then, by (i), we know that no solution exists. If d | y, then by (ii) we need only solve the equation x · (g/d) = (y/d) mod (N/d). By (iii), we have gcd(g/d, N/d) = 1 and so we can compute the solution x = [(y/d) · (g/d)−1 mod (N/d)]. 9.4 Here we show how to solve the discrete logarithm problem in a cyclic √ group of order q = pe in time O(polylog(q)· p). Given as input a generator g of known order pe and a value y, we want to compute x = logg y. √ (a) Show how to find [x mod p] in time O(polylog(q) · p). (b) Say x = x0 +x1 ·p+· · ·+xe−1 ·pe−1 with 0 ≤ xi < p. In the previous step we determined x0 . Show how to compute in polylog(q) time a e−2 value y1 such that (g p)x1 +x2 ·p+···+xe−1 ·p = y1 . (c) Use recursion to obtain the claimed running time for the original problem. (Note that e = log(q).) Solution: e−1 e−1 (a) As suggested by the hint, set g0 := g p and y0 := y p . Note x that g0 = y0 , and so there must exist an x0 ∈ {0, . . . , ord(g0) − 1} such that g0x0 = y0 . Since ord(g0) = pe /pe−1 = p (cf. Lemma 8.7), it follows from Proposition 7.50 that x0 = [x mod p]. Working in the group of order p generated by g0 , we can compute x0 in time √ p using, e.g, the baby-step/giant-step algorithm. (b) Setting y1 := y · g −x0 works. Thus is so since y · g −x0 = g x0 +x1 ·p+···+xe−1 ·p e−1 x1 +x2 ·p+···+xe−1 ·p = (g p) · g −x0 e−2 . (c) Note that ord(g p) = pe−1 , and so the group G1 generated by g p has def order pe−1 . Letting x′ = x1 +x2 ·p+· · ·+xe−1 ·pe−2 , we can apply the result from part (a) in group G1 to compute [x′ mod p] = x1 . Repeatedly applying parts (b) and (a) thus allows us to compute x2 , . . . , xe−1 and hence all of x. The computation requires a total of √ O(e) = O(log q) computations, each taking time O(polylog(q)· p), giving the claimed running time. Factoring and Computing Discrete Logarithms 103 ∏k 9.5 Let q have prime factorization q = i=1 pei i . Using the result from the previous problem, show a modification of the Pohlig-Hellman algorithm that discrete logarithm problem in a group of order q in time (solves the ∑ ( ) √) k O polylog(q) · i=1 ei pi = O polylog(q) · max{ pi } . Solution: Using the Pohlig-Hellman algorithm, a discrete-logarithm computation in a group of order q reduces to k discrete-logarithm computations in groups of order pe11 , . . . , pekk . The previous exercise shows that discrete logarithms in a group of order pe can be computed in time √ O(polylog(pe) · p). Thus, the total computation is (∑ √) k ei O polylog(p) · pi . i i=1 ei Since k =) O(log q), this simplifies to a complexity of (pi = O(q) and √ O polylog(q) · maxi { pi } . 9.6 Give pseudocode for the small-space algorithm for computing discrete logarithms described in Section 9.2.3, and give a heuristic analysis of the probability with which it succeeds. Solution: Let g, h ∈ G be given; we want to compute logg h. To do this, define F : G → Zq × Zq (modeled as a random function), Hg,h : Zq × Zq → G, and H = Hg,h ◦ F : G → G as in Section 9.2.3. √ Use Algorithm 5.9 to find a collision k, k ′ in H using q invocations √ of H, and hence q invocations of F . With overwhelming probability, F (k) ≠ F (k ′); this follows from Lemma A.15 because the range of F has size q 2 . Thus, F (k), F (k ′) are a collision in Hg,h , and as discussed in Theorem 8.79 allow computation of logg h. Chapter 10 Key Management and the Public-Key Revolution – Solutions 10.1 Let Π be a key-exchange protocol, and (Enc, Dec) be a private-key encryption scheme. Consider the following interactive protocol Π′ for encrypting a message: first, the sender and receiver run Π to generate a shared key k. Next, the sender computes c ← Enck (m) and sends c to the other party, who decrypts and recovers m using k. (a) Formulate a definition of indistinguishable encryptions in the presence of an eavesdropper (cf. Definition 3.8) appropriate for this interactive setting. (b) Prove that if Π is secure in the presence of an eavesdropper and (Enc, Dec) has indistinguishable encryptions in the presence of an eavesdropper, then Π′ satisfies your definition. Solution: (a) Consider the following experiment, defined for any interactive protocol Π′ (that is in turn composed of a key-exchange protocol Π and a private-key encryption scheme Πenc = (Gen, Enc, Dec)), any adversary A, and any value n for the security parameter: The eavesdropping indistinguishability experiment KE-EncA,Π,Πenc (n): (a) The adversary A is given input 1n , and outputs a pair of messages m0 , m1 of the same length. (b) Two parties holding 1n execute protocol Π, which results in a transcript trans and a key k that is output by each of the parties. (c) A random bit b ← {0, 1} is chosen, and then a ciphertext c ← Enck (mb) is computed. (d) A is given trans and c, and outputs a bit b′ . (e) The output of the experiment is defined to be 1 if b′ = b, and 0 otherwise. DEFINITION Interactive protocol Π′ , composed of Π and Πenc , has indistinguishable encryptions in the presence of an eavesdropper if for all 105 106 Introduction to Modern Cryptography – 2nd Edition Solutions Manual probabilistic polynomial-time adversaries A there exists a negligible function negl such that Pr [KE-EncA,Π,Πenc (n) = 1] ≤ 1 + negl(n), 2 where the probability is taken over the random coins used by A, as well as the random coins used in the experiment. (b) Fix an arbitrary probabilistic polynomial-time adversary A. Our goal is to prove that there exists a negligible function negl such that Pr [KE-EncA,Π,Πenc (n) = 1] ≤ 1 + negl(n). 2 Let us write Pr [KE-EncA,Π,Πenc (n) = 1] = Pr (trans,k)←Π [A(trans, Enck (mb)) = b], where b is chosen uniformly at random, trans is generated by running an execution of protocol Π, and k is the key resulting from that execution. (To emphasize the latter, we include (trans, k) ← Π as a subscript.) We first prove that there is a negligible function negl1 such that Pr [A(trans, Enck (mb)) = b] (10.1) (trans,k)←Π ≤ Pr (trans,k)←Π [A(trans, Enck′ (mb)) = b] + negl1 (n), where k ′ is now chosen uniformly at random, independently of trans. The above follows from the assumed security of Π. We then show that there exists a negligible function negl2 with Pr (trans,k)←Π [A(trans, Enck′ (mb)) = b] ≤ 1 + negl2 (n). 2 (10.2) This follows from security of Πenc . Taken together, Equations (10.1) and (10.2) prove the desired result. We now prove Equation (10.1). Consider the following adversary A1 that eavesdrops on an execution of Π: Adversary A1 : ˆ Let n = |k|. ˆ (a) A1 is given a transcript trans and a key k. n (b) A1 runs A(1) to obtain two messages (m0 , m1). (c) A1 chooses a bit b ← {0, 1} and computes c ← Enckˆ (mb). It then runs A(trans, c) to obtain a bit b′ . (d) If b′ = b then A1 outputs 1; otherwise, A1 outputs 0. Key Management and the Public-Key Revolution 107 Consider the experiment KEeav A1 ,Π (n). When the random bit (call it bke) used in that experiment is equal to 1, then kˆ is equal to the actual key generated by the execution of Π resulting in trans. Furthermore, in this case the experiment evaluates to 1 exactly when A1 outputs 1 (since this implies that A1 has correctly guessed the value of bke = 1). Since this occurs exactly when A correctly guesses the value of b, we have Pr[KEeav A1 ,Π (n) = 1 | bke = 1] = Pr [A(trans, Enck (mb)) = b]. (trans,k)←Π On the other hand, when bke = 0 then kˆ is a uniformly random key, chosen independently of anything else. In this case, the experiment evaluates to 1 if and only if A1 outputs 0. Since this occurs only when A correctly fails to guess the value of b, we have [ ] Pr KEeav Pr [A(trans, Enck′ (mb)) = b] . A1 ,Π (n) = 1 | bke = 0 = 1 − (trans,k)←Π Altogether, then, we have Pr[KEeav A1 ,Π (n) = 1] 1 1 = · Pr[KEeav · Pr[KEeav A1 ,Π (n) = 1 | bke = 1] + A1 ,Π (n) = 1 | bke = 0] 2 2 (1 1 Pr [A(trans, Enck (mb)) = b] = + · 2 2 (trans,k)←Π) − Pr [A(trans, Enck′ (mb)) = b] . (trans,k)←Π But since Π is secure in the presence of an eavesdropper, we know there exists a negligible function negl′1 such that Pr[KEeav A1 ,Π (n) = 1] ≤ 1 + negl′1 (n). 2 These last two equations imply Equation (10.1). We now prove Equation (10.2). Consider the following adversary A2 , attacking Πenc in the sense of Definition 3.8: Adversary A2 : (a) A2 is given 1n , and runs A(1n) to obtain two messages (m0 , m1). These same messages are output by A2 . (b) A2 is given a ciphertext c. (c) A2 runs Π itself, playing the roles of both Alice and Bob, to generate a transcript trans and a key k. (It will ignore k.) 108 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (d) A2 runs A(trans, c) to obtain a bit b. This same bit is output by A2 . ′ In experiment PrivKeav A2 ,Πenc (n), the key k used (by the experiment) to generate the ciphertext c is indeed chosen uniformly and independently of trans. Furthermore, A is given an encryption of m0 (resp., m1) exactly when A2 is given an encryption of m0 (resp., m1), and A2 outputs as its guess whatever is output by A. Thus, Pr[PrivKeav A2 ,Πenc (n) = 1] = Pr [A(trans, Enck′ (mb)) = b]. (trans,k)←Π Since Πenc satisfies Definition 3.8, we know that there exists a negligible function negl2 such that Pr[PrivKeav A2 ,Πenc (n) = 1] ≤ 1 + negl2 (n). 2 Equation (10.2) follows. 10.2 Show that, for either of the groups considered in Sections 8.3.3 or 8.3.4, a uniform group element (expressed using the natural representation) is easily distinguishable from a uniform bit-string of the same length. Solution: Consider first the case of a subgroup G of order q of Z∗p , with p = rq + 1. As we have seen in Section 8.3.3, h ∈ G if and only if hq = 1 mod p. Now, a uniform string of the same length will be in G with probability at most 1/r, and so can easily be distinguished. However, a much more serious problem arises with the higher order bits of a uniform string. To be concrete, let p = 23, with a representation of 5 bits. Then, a uniform 5-bit string will be greater than or equal to p with probability 9/32, whereas an element of any subgroup of Z∗32 will never be greater than or equal to p. Thus, elements of the group will have the most significant bit equal to 1 with probability significantly lower than a uniform string. In Section 8.3.4, elements of the group are pairs (x, y) that satisfy the equation y 2 = x3 + Ax + B mod p. The number of elements on the def curve is q = |E(Zp)|, whereas the number of pairs (x, y) is 22n (where n denotes the number of bits in the representation of p). By Theorem 8.70, √ we

have ε q p n + 1 + 2 p ≪ 22n . Thus, the vast majority of pairs (x, y) will not satisfy the equation, and a uniform string can be easily distinguished from a uniform group element. 10.3 Describe a man-in-the-middle attack on the Diffie–Hellman protocol where the adversary shares a key kA with Alice and a (different) key kB with Bob, and Alice and Bob cannot detect that anything is wrong. Solution: The man-in-the-middle attack consists of simply running independent executions of the protocol with Alice and with Bob. (I.e., Key Management and the Public-Key Revolution 109 the adversary plays the role of Bob when interacting with Alice, and plays the role of Alice when interacting with Bob.) This results in a key kA output by Alice and a key kB output by Bob, both of which are known to the adversary. Since the adversary ran the protocol honestly with each party, neither party can detect that anything is amiss. 10.4 Consider the following key-exchange protocol: (a) Alice chooses uniform k, r ∈ {0, 1}n , and sends s := k ⊕ r to Bob. (b) Bob chooses uniform t ∈ {0, 1}n , and sends u := s ⊕ t to Alice. (c) Alice computes w := u ⊕ r and sends w to Bob. (d) Alice outputs k and Bob outputs w ⊕ t. Show that Alice and Bob output the same key. Analyze the security of the scheme (i.e., either prove its security or show a concrete attack). Solution: Alice outputs k, while Bob outputs w ⊕ t = (u ⊕ r) ⊕ t = ((s ⊕ t) ⊕ r) ⊕ t = (((k ⊕ r) ⊕ t) ⊕ r) ⊕ t = k. The scheme, however, is not secure. Given a transcript (s, u, w) of an execution of the protocol, an eavesdropper can compute s ⊕ u ⊕ w and this is equal to the key since s ⊕ u ⊕ w = (k ⊕ r) ⊕ (k ⊕ r ⊕ t) ⊕ (k ⊕ t) = k. Chapter 11 Public-Key Encryption – Solutions 11.1 Assume a public-key encryption scheme for single-bit messages with no decryption error. Show that, given pk and a ciphertext c computed via c ← Encpk (m), it is possible for an unbounded adversary to determine m with probability 1. Solution: This is just a trivial brute-force search. We describe the attack assuming that m is a single bit, but the attack generalizes easily to m of arbitrary length or even m of unknown length. The attack is as follows: let ℓ = ℓ(n) denote the length of the random coins used by Enc. Given pk and c, compute cr := Encpk (0; r) for all r ∈ {0, 1}ℓ . If c = cr for some r, then output "m = 0"; else, output "m = 1." This adversary is clearly far from efficient, but that is ok since in this exercise we allow the adversary to be unbounded. If c is an encryption of 0 then c = Encpk (0; r) for some r and so the above adversary always (correctly) outputs "m = 0." But if c is an encryption of 1 then c ≠ cr for all r. (This follows from correctness.) So in this case the adversary always (correctly) outputs "m = 1." 11.2 Show that for any CPA-secure public-key encryption scheme for singlebit messages, the length of the ciphertext must be superlogarithmic in the security parameter. Solution: We aim for simplicity rather than trying to optimize parameters. Let Π = (Gen, Enc, Dec) be an encryption scheme. For a given public key pk generated using Gen and a bit b, let Cpk (b) denote the set of ciphertexts corresponding to possible encryptions of b with respect to pk. Say that for some fixed bit b the length of the ciphertext after encrypting b is at most logarithmic in the security parameter (for every public key). Then for any public key, the set Cpk (b) has size at most p(n) for some polynomial p. We exploit this in the following attack: Adversary A: (a) A is given pk, and computes c0 ← Encpk (0) and c1 ← Encpk (1). (b) A outputs the two messages (0, 1), and is given in return a ciphertext c. 111 112 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) If c = c0 then A outputs 0; if c = c1 then A outputs 1; otherwise A outputs a random bit. As above, let b be the bit for which |Cpk (b)| ≤ p(n) (if this holds for both 0 and 1 then set b arbitrarily). Let c∗b ∈ Cpk (b) be the ciphertext that occurs most frequently when encrypting b, and note that Pr[Encpk (b) = c∗b] ≥ 1/|Cpk (b)|. We now analyze the behavior of A. Pr[A outputs b | c is an encryption of b] = Pr[c = cb] + 1 · Pr[c′ = cb] 2 1 1 ∧ 1 1 = + · Pr[c = c∗b cb = c∗b] 2 2 1 1 ≥ + · 2 2p (n) = (In fact, one can show that Pr[c = cb] ≥ 1/|Cpk (b)| ≥ 1/p(n) but this is not needed for the proof.) We also have Pr[A outputs ¯b | c is an encryption of ¯b] 1 1 = Pr[c = c¯b] + · Pr[c′ = c¯b] ≥ . 2 2 From the above, we have 1 · Pr[A outputs b | c is an encryption of b] 2 1 + · Pr[A outputs ¯b | c is an encryption of ¯b] 2( ) 1 1 1 1 1 1 ≥ · + + · = + . 2 2 2p2 (n) 2 2 2 4p2 (n) Pr[PubKeav A,Π (n) = 1] = Since 1/4p2 (n) is not negligible, Π is not CPA-secure. 11.3 Say a public-key encryption scheme (Gen, Enc, Dec) for n-bit messages is one-way if any ppt adversary has negligible probability of success in the following experiment: • Gen(1n) is run to obtain keys (pk, sk). • A message m ∈ {0, 1}n is chosen uniformly at random, and a ciphertext c ← Encpk (m) is computed. • A is given pk and c, and outputs a message m′ . We say A succeeds if m′ = m. (a) Give a construction of a CPA-secure KEM in the random-oracle model based on any one-way public-key encryption scheme. Public-Key Encryption 113 (b) Can a deterministic public-key encryption scheme be one-way? If not, prove impossibility; if so, give a construction based on any of the assumptions introduced in this book. Solution: (a) See Construction 11.1-S. CONSTRUCTION 11.1-S Let Π = (Gen, Enc, Dec) be a public-key encryption scheme, and let ℓ(n) be an arbitrary polynomial. Let H be a function whose domain is {0, 1}∗ and whose range can be set to {0, 1}ℓ(n) for any n. Construct a public-key encryption scheme Π′ as follows: • Gen′ : on input 1n , run Gen(1n) to obtain (pk, sk). The public key is pk and the private key is sk. • Enc′ : on input a public key pk and a message m ∈ {0, 1}ℓ(n) , choose a random r ← {0, 1}n and output the ciphertext ⟨Encpk (r), H(r) ⊕ m⟩ . • Dec′ : on input a private key sk and a ciphertext ⟨c1 , c2 ⟩, compute r := Decsk (c1) and then output the message H(r) ⊕ c2 . A proof of security is similar-in-spirit to the proof of Theorem 11.38, though simpler because only CPA-security is being shown. (b) A public-key encryption scheme where encryption is deterministic can be one-way. Textbook RSA encryption (Construction 11.26) serves as one example: this scheme is one-way as long as the RSA problem is hard relative to GenRSA. 11.4 Show that any two-round key-exchange protocol (that is, where each party sends a single message) satisfying Definition 10.1 can be converted into a CPA-secure public-key encryption scheme. Solution: Let Π be a 2-round key-exchange protocol where Alice goes first. A public-key encryption scheme can be constructed as in Construction 11.2-S. Correctness follows from the fact that msg1 , msg2 jointly form an execution of Π, and so the sender and receiver generate the same key k. Thus, the receiver obtains the sender's intended message m. Let Πenc denote Construction 11.2-S. We prove that Πenc has indistinguishable encryptions in the presence of an eavesdropper. Let A be a probabilistic, polynomial-time adversary, and define def ε(n) = Pr[PubKeav A,Πenc (n) = 1]. 114 Introduction to Modern Cryptography – 2nd Edition Solutions Manual CONSTRUCTION 11.2-S Let Π be as in the text. Define a public-key encryption scheme as follows: • Gen: on input 1n run Alice's side of Π to obtain msg1 along with some state information s. The public key is msg1 and the private key is s. • Enc: on input public key pk = msg1 and message m ∈ {0, 1}n , the sender runs Bob's side of Π to obtain msg2 and a key k ∈ {0, 1}n. Output the ciphertext ⟨msg2 , k ⊕ m⟩. • Dec: on input a private key sk = s and a ciphertext ⟨msg2 , c⟩, run Alice's side of Π (using state s and incoming message msg2) to obtain a key k. Then output the message m := c ⊕ k. Public-key encryption from key exchange. e where Gen is the same Consider the modified "encryption scheme" Π as in Πenc , but encryption of a message m with respect to the public key msg1 is done by running Bob's side of Π to obtain msg2 but then choosing a random k ∈ {0, 1}n and outputting the ciphertext ⟨msg2 , k ⊕ m⟩. e is not actually an encryption scheme (as there is no way Although Π for the receiver to decrypt), the experiment PubKeav e (n) is still wellA,Π defined since that experiment depends only on the key generation and encryption algorithms. As in the case of the one-time pad, the second component of the ciphere is a uniform string and, in particular, is independent of text in scheme Π the message m being encrypted. The first component of the ciphertext is also independent of m. It follows that Pr[PubKeav e (n) = 1] = A,Π 1 . 2 Now consider the following ppt algorithm D attacking Π: Algorithm D: The algorithm is given trans = (msg1 , msg2), k as input. • Set pk = msg1 and run A(pk) to obtain two messages m0 , m1 . • Choose a uniform bit b, and give ⟨msg2 , k ⊕ mb ⟩ to A. • When A outputs b′ , output 1 if b′ = b and 0 otherwise. Let us analyze the behavior of D. There are two cases to consider: Public-Key Encryption 115 Case 1: Say the input to D is generated by running Π to obtain a transcript trans and a key k. In this case the view of A when run as a subroutine by D is distributed identically to A's view in experiment PubKeav A,Πenc (n), and so D outputs 1 in this case with probability Pr[PubKeav A,Πenc (n) = 1]. Case 2: Say the input to D is generated by running Π to obtain a transcript trans and a key k′ , but then choosing k uniformly from {0, 1}n and giving D the pair (trans, k). In this case the view of A when run as a subroutine by D is distributed identically to A's view in experiment ′ PubKeav e (n). Since D outputs 1 exactly when the output b of A is A,Π equal to b, the probability that D outputs 1 in this case is exactly 1 Pr[PubKeav e (n) = 1] = 2 . A,Π Since Π is a secure key-exchange protocol, there exists a negligible function negl such that 1 + negl(n) ≥ Pr[KEeav A,Π (n) = 1] 2 1 = · Pr[D outputs 1 | k is output by Π] 2 1 + · Pr[D outputs 0 | k is random] 2 1 1 1 · . = · Pr[PubKeav A,Πenc (n) = 1] + 2 2 2 We thus conclude that Pr[PubKeav A,Πenc (n) = 1] ≤ 1 + 2 · negl(n). 2 Since A was arbitrary, this proves that Πenc has indistinguishable encryptions in the presence of an eavesdropper. 11.5 Show that Claim 11.7 does not hold in the setting of CCA-security. Solution: Let Π = (Gen, Enc, Dec) be a CCA-secure encryption scheme for 1-bit messages, and let Π′ = (Gen, Enc′ , Dec′) denote the encryption scheme for arbitrary-length messages where Encpk (m1 · · · mt) = Encpk (m1), . . . , Encpk (mt). We show that Π′ is not CCA-secure. Consider the following adversary: Algorithm A: (a) Given pk, output messages m0 = 00 and m1 = 11. (b) Receive in return a ciphertext c = c1 , c2 . 116 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) Compute c′2 ← Encpk (0). Submit the ciphertext c′ = c1 , c′2 to the decryption oracle, and receive in return the message b 0, with b ∈ {0, 1}. (d) Output b. First note that this is a legal attack: m0 and m1 have the same length, and the ciphertext c′ submitted to the decryption oracle satisfies c′ ≠ c. It is easy to see that this attacker succeeds in guessing the bit b used by the experiment with probability 1; thus, Π′ is not CCA-secure. 11.6 Consider the following public-key encryption scheme. The public key is (G, q, g, h) and the private key is x, generated exactly as in the El Gamal encryption scheme. In order to encrypt a bit b, the sender does the following: (a) If b = 0 then choose a uniform y ∈ Zq and compute c1 := g y and c2 := hy . The ciphertext is ⟨c1 , c2 ⟩. (b) If b = 1 then choose independent uniform y, z ∈ Zq , compute c1 := g y and c2 := g z , and set the ciphertext equal to ⟨c1 , c2 ⟩. Show that it is possible to decrypt efficiently given knowledge of x. Prove that this encryption scheme is CPA-secure if the decisional Diffie– Hellman problem is hard relative to G. Solution: Decryption is easy. Given a ciphertext ⟨c1 , c2 ⟩ and secret key x, output 0 iff cx1 = c2 . When the ciphertext is an encryption of 0 then decryption is always correct since x y cx1 = (g y) = (g x) = hy = c2 . When the ciphertext is an encryption of 1 then cx1 = g yx . Unless it happens that z = xy, decryption will correctly output 1. But z = xy only with probability 1/q, which is negligible. Let Π denote the encryption scheme of the exercise. Security of Π follows fairly immediately from the observation that when ⟨c1 , c2 ⟩ is an encryption of 0 then c2 = DHg (h, c1), whereas if ⟨c1 , c2 ⟩ is an encryption of 1 then c1 and c2 are independent, random elements of G. Formally, given an adversary A attacking the encryption scheme we can construct an algorithm A′ solving the DDH problem as follows: Algorithm A′ : The algorithm is given G, q, g, h, c1 , c2 as input. (a) Run A on the public key pk = ⟨G, q, g, h⟩. Assume for simplicity that A outputs (m0 = 0, m1 = 1). (b) Give to A the ciphertext ⟨c1 , c2 ⟩. Output whatever bit b′ is output by A. Public-Key Encryption 117 Consider the case when the input to A′ is generated by running G(1n) to obtain (G, q, g), then choosing uniform x, y, z ∈ Zq , and finally setting h := g x , c1 := g y , and c2 := g z . Then, from the point of view of A (run as a sub-routine by A′), the experiment is equivalent to PubKeav A,Π (n) conditioned on b = 1 (i.e., conditioned on the ciphertext ⟨c1 , c2 ⟩ being an encryption of 1). Furthermore, A′ outputs 1 in this case exactly when A outputs 1, i.e., when A succeeds. Thus: Pr[A′ (G, q, g, g x , g y , g z) = 1] = Pr[PubKeav A,Π (n) = 1 | b = 1]. When the input to A′ is generated by running G(1n) to obtain (G, q, g), then choosing uniform x, y ∈ Zq , and finally setting h := g x , c1 := g y , and c2 := g xy then, from the point of view of A, the experiment is ′ equivalent to PubKeav A,Π (n) conditioned on b = 0. Again, A outputs 1 when A does; now, however, this means that A did not succeed. So, Pr[A′ (G, q, g, g x , g y , g xy) = 1] = 1 − Pr[PubKeav A,Π (n) = 1 | b = 0]. Putting this together and assuming the DDH problem is hard relative to G, there exists a negligible function negl such that negl(n) ≥ |Pr[A′ (G, q, g, g x , g y , g z) = 1] − Pr[A′ (G, q, g, g x , g y , g xy) = 1]| eav = Pr[PubKeav A,Π (n) = 1 | b = 1] + Pr[PubKA,Π (n) = 1 | b = 0] − 1 1 = 2 · Pr[PubKeav A,Π (n) = 1] − . 2 Since A was arbitrary, it follows that Π is CPA-secure. 11.7 Consider the following variant of El Gamal encryption. Let p = 2q + 1, let G be the group of squares modulo p (so G is a subgroup of Z∗p of order q), and let g be a generator of G. The private key is (G, g, q, x) and the public key is (G, g, q, h), where h = g x and x ∈ Zq is chosen uniformly. To encrypt a message m ∈ Zq , choose a uniform r ∈ Zq , compute c1 := g r mod p and c2 := hr + m mod p, and let the ciphertext be ⟨c1 , c2 ⟩. Is this scheme CPA-secure? Prove your answer. Solution: This scheme is not secure. In particular, consider an adversary A that chooses uniform m0 , m1 ∈ Zq and then receives the challenge ciphertext ⟨c1 , c2 ⟩. Observe that since c2 = hy + m mod p it is not necessarily the case that c2 ∈ G (since addition is not the group operation). However, it is guaranteed that [c2 − mb mod p] is in G, where mb is the value that was encrypted in ⟨c1 , c2 ⟩. Furthermore, as we have seen in the text, it is possible to efficiently verify if a value is in such a group G. The question remains as to the probability that [c2 −m1−b mod p] is also in G. However, as we know, G includes exactly 118 Introduction to Modern Cryptography – 2nd Edition Solutions Manual half of the elements of Z∗p . Thus, since m1−b is a random value, it follows that [c2 − m1−b mod p] ∈ G with probability only 1/2. Thus, A's strategy is to check if [c2 − m0 mod p] ∈ G and if [c2 − m1 mod p] ∈ G. If both elements are in the group, then A outputs a uniform bit. Otherwise, it outputs the bit b for which [c2 − mb mod p] ∈ G. A simple calculation shows that A succeeds with probability 3/4. 11.8 Consider the following protocol for two parties A and B to flip a fair coin (more complicated versions of this might be used for Internet gambling): (1) a trusted party T publishes her public key pk; (2) then A chooses a uniform bit bA , encrypts it using pk, and announces the ciphertext cA to B and T ; (3) next, B acts symmetrically and announces a ciphertext cB ≠ cA ; (4) T decrypts both cA and cB , and the parties XOR the results to obtain the value of the coin. (a) Argue that even if A is dishonest (but B is honest), the final value of the coin is uniformly distributed. (b) Assume the parties use El Gamal encryption (where the bit b is encoded as the group element g b before being encrypted—note that efficient decrypt is still possible). Show how a dishonest B can bias the coin to any value he likes. (c) Suggest what type of encryption scheme would be appropriate to use here. Can you define an appropriate notion of security and prove that your suggestion achieves this definition? Solution: (a) The ciphertext cA output by A corresponds to some bit bA . (It could also be an invalid ciphertext that does not decrypt at all, though this cannot occur [without being detected] when using El Gamal encryption. If T detects an invalid ciphertext upon decryption, then it can just use cB as the coin.) If B is honest, then it chooses a uniform cB , and so the coin cA ⊕ cB (or even just cB in case cA is invalid) is uniform. (b) The problem when trying to apply the argument in part (a) to the case when A is honest but B is not is that B might choose cB in a way that depends on cA (and hence bA); this was not possible when the dishonest party went first. We show that, when El Gamal encryption is used, B can bias the coin any way it likes. B can bias the coin toward 0 as follows. Given the ciphertext cA = ⟨c1 , c2 ⟩ broadcast by A, player B chooses a random r ← Zq and outputs cB = ⟨c′1 , c′2 ⟩ with c′1 := c1 · g r and c′2 := c2 · hr . Public-Key Encryption 119 One can check that cB always decrypts to bA , and so the coin is always 0 if B follows this strategy. Furthermore, it holds that cB is distributed exactly like a random encryption of bA (this is not too much more difficult to show), and so this kind of cheating by B is undetectable (since it is indistinguishable whether B is cheating or whether it just happened that B chose bB = bA). To bias toward 1, player B can do the following. Given cA as above, choose random r ← Zq and output cB = ⟨c′1 , c′2 ⟩ with −1 c′1 := (c1) · g r and c′2 := (c2) −1 · g · hr . One can check that now bB = 1 − bA , and so the coin always comes out to be 1. Once again, cB is in fact a random encryption of 1−bA and so this form of cheating is undetectable. (c) A CCA-secure encryption scheme should be used to prevent the problems sketched in part (b). We define security informally, since security of two-party protocols is not covered in the book. Say a coin-tossing protocol Π is secure if, for any ppt adversary corrupting either one of the parties, the probability p = p(n) that the resulting coin is 0 is negligibly close to 12 . (I.e., there is a negligible function negl such that |p(n) − 12 | ≤ negl(n).) Note that we use absolute value here since we don't want the coin to be biased toward either 0 or 1. We claim that the protocol in this exercise, when instantiated with a CCA-secure public-key encryption scheme, is a secure coin-tossing protocol. (We add one point of clarification for the protocol: if either party's ciphertext is invalid, including the case when it does not decrypt to 0/1, then T outputs the bit obtained by decrypting the other party's ciphertext.) We have seen already in part (a) that an adversary corrupting A cannot bias the coin. So consider the case when B is corrupted. Let Π denote the CCA-secure encryption scheme used in the protocol. Define the following adversary A attacking Π: Adversary A: i. A is given pk, and outputs m0 = 0 and m1 = 1. ii. A is given in return a ciphertext cA . It then runs B(cA) to obtain a ciphertext cB ≠ cA . A submits cB to its decryption oracle and receives in return a bit b′ . Then: • If decryption of cB returns an error, or if b′ ∉ {0, 1} then A outputs 0. • If b′ = 0 then A outputs 0. • If b′ = 1 then A outputs 1. 120 Introduction to Modern Cryptography – 2nd Edition Solutions Manual We have 1 · Pr[A outputs 0 | cA is an encryption of 0] 2 1 + · Pr[A outputs 1 | cA is an encryption of 1] 2 1 = · Pr[coin=0 | cA is an encryption of 0] 2 1 + · Pr[coin=0 | cA is an encryption of 1] 2 Pr[PubKcca A,Π (n) = 1] = def = Pr[coin=0] = p(n). CCA-security of Π implies that p(n) ≤ 21 + negl(n) for some negligible function negl. A similar argument, flipping the output of A, shows that 1−p(n) ≤ 12 +negl(n) as well. Together, these complete the proof of security for the protocol. 11.9 Prove formally that the El Gamal encryption scheme is not CCA-secure. Solution: Consider the following attacker A: given public key (G, q, g, h), choose arbitrary (distinct) h0 , h1 ∈ G as the challenge messages. Upon receiving the challenge ciphertext ⟨c1 , c2 ⟩, compute c′2 = h · c2 and request decryption of ⟨c1 , c′2 ⟩, receiving some u in return. Finally, output 0 if uh = h0 and output 1 if uh = h1 . We prove that A succeeds in the CCA-experiment with probability 1. Observe that ⟨c1 , c2 ⟩ = ⟨g y , hy · hb ⟩ and so ⟨c1 , c′2 ⟩ = ⟨g y , hy · hb · h⟩. This implies that the decryption oracle returns u = hb · h and thus u h = hb , as required. 11.10 In Section 11.4.4 we showed that El Gamal encryption is malleable, and specifically that given a ciphertext ⟨c1 , c2 ⟩ that is the encryption of some unknown message m, it is possible to produce a ciphertext ⟨c1 , c′2 ⟩ that is the encryption of α · m (for known α). A receiver who receives both these ciphertexts might be suspicious since both ciphertexts share the first component. Show that it is possible to generate ⟨c′1 , c′2 ⟩ that is the encryption of α · m, with c′1 ≠ c1 and c′2 ≠ c2 . def Solution: Let ⟨c1 , c2 ⟩ = ⟨g y , hy · m⟩. Then, for any r we have ⟨c′1 , c′2 ⟩ = ⟨c1 · g r , c2 · hr · α⟩ = ⟨g y+r , hy+r · α · m⟩. Thus, ⟨c′1 , c′2 ⟩ is an encryption of α·m with c′1 ≠ c1 and c′2 ≠ c2 . (In fact, c′1 is uniform and independent of c1 and ⟨c′1 , c′2 ⟩ is an independent random encryption of α · m.) 11.11 Prove Theorem 11.22. Solution: The proof is similar to that of Theorem 11.38, and is omitted. 11.12 One of the attacks on plain RSA discussed in Section 11.5.1 involves a sender who encrypts two related messages using the same public key. Public-Key Encryption 121 Formulate an appropriate definition of security ruling out such attacks, and show that any CPA-secure public-key encryption scheme satisfies your definition. Solution: There are multiple ways a definition can be formulated; we provide one. Consider the following experiment involving an attacker A and an encryption scheme Π: PubKrelated A,Π (n): (a) Gen(1n) is run to obtain keys (pk, sk). (b) Adversary A is given pk, and outputs a pair of messages m0 , m1 and a function f . We require that |m0 | = |m1 | and |f (m0)| = |f (m1)| and that m0 , m1 , f (m0), f (m1) are in the message space. (c) A uniform bit b ∈ {0, 1} is chosen, and then ciphertexts c ← Encpk (mb) and c′ ← Encpk (f (mb)) are computed and given to A. (d) A outputs a bit b′ . The output of the experiment is 1 if b′ = b, and 0 otherwise. Π = (Gen, Enc, Dec) is secure for encryption of related messages if for all probabilistic, polynomial-time adversaries A there is a negligible function negl such that Pr[PubKrelated A,Π (n) = 1] ≤ negl(n). Any attacker in the above experiment can be simulated by an attacker in the PubKLR-cpa experiment (by submitting (m0 , m1) to the LR-oracle, followed by (f (m0), f (m1))). Since any CPA-secure scheme is secure in that experiment, this implies that any CPA-secure scheme is also secure for encryption of related messages according to the definition above. 11.13 One of the attacks on plain RSA discussed in Section 11.5.1 involves a sender who encrypts the same message to three different receivers. Formulate an appropriate definition of security ruling out such attacks, and show that any CPA-secure public-key encryption scheme satisfies your definition. Solution: There are multiple ways a definition can be formulated; we provide one. Consider the following experiment involving an attacker A and an encryption scheme Π: -keys (n): PubKℓA,Π (a) Gen(1n) is run ℓ = ℓ(n) times to obtain (pk1 , sk1), . . . , (pkℓ , skℓ). (b) A is given pk1 , . . . , pkℓ , and outputs a pair of equal-length messages m0 , m1 . 122 Introduction to Modern Cryptography – 2nd Edition Solutions Manual (c) A uniform b ∈ {0, 1} is chosen, and (for i = 1, . . . , ℓ) ciphertexts ci ← Encpki (mb) are computed and given to A. (d) A outputs a bit b′ . The output of the experiment is 1 if b′ = b, and 0 otherwise. Π = (Gen, Enc, Dec) is secure for encrypting to different receivers if for all polynomials ℓ and all ppt adversaries A there is a negligible function negl such that -keys (n) = 1] ≤ negl(n). Pr[PubKℓA,Π A proof that CPA-security of Π implies security for encrypting to different receivers uses a hybrid argument as in the proof of Theorem 11.6, and is omitted. 11.14 Consider the following modified version of padded RSA encryption: Assume messages to be encrypted have length exactly ‖N ‖ /2. To encrypt, first compute m ˆ := 0x00‖r‖0x00‖m where r is a uniform string of length ‖N ‖ /2 − 16. Then compute the ciphertext c := [m ˆ e mod N]. When decrypting a ciphertext c, the receiver computes m ˆ := [cd mod N] and returns an error if m ˆ does not consist of 0x00 followed by ‖N ‖ /2 − 16 arbitrary bits followed by 0x00. Show that this scheme is not CCAsecure. Why is it easier to construct a chosen-ciphertext attack on this scheme than on PKCS #1 v1.5? Solution: Consider the following adversary: Algorithm A: (a) Given the public key N, e, output m0 = 00 · · · 0 and m1 = 01 · · · 1. Receive in return a ciphertext c. (b) Set c′ := [2e · c mod N] and submit c′ to the decryption oracle. (c) If the decryption oracle returns an error, then output a random bit. Otherwise, let m′ denote the message returned by the decryption oracle. (d) Output the most-significant bit of m′ . Why does this attack succeed (with high probability)? Let b denote the bit such that c is an encryption of mb , and let k = ‖N ‖ /2 − 16. Then c = [(0x00 ‖ r1 · · · rk ‖ 0x00 ‖ 0b · · · b)e mod N] and c′ = 2e · c = [(2 · (0x00 ‖ r1 · · · rk ‖ 0x00 ‖ 0b · · · b))e mod N] = [(07 r1 ‖ r2 · · · rk 0 ‖ 0x00 ‖ b · · · b0)e mod N]. (Multiplication by 2 corresponds to a left-wise bit shift; there is no overflow since the leading bit of the multiplicand is 0.) Decryption of Public-Key Encryption 123 c′ results in an error only if r1 = 1, which occurs with probability 21 ; if no error occurs, then the high-order bit of the decrypted message is exactly equal to b. Thus the above attack correctly determines b with probability 3/4, which is significantly better than guessing at random. This simple attack does not work on PKCS #1 v1.5. In that scheme, encryption of a message m is done by computing c := [(00000000 ‖ 00000010 ‖ r1 r2 · · · ‖ 00000000 ‖ m1 m2 · · ·)e mod N] , where r is uniform. Thus, setting c′ := [2e · c mod N] would give c′ = [(00000000 ‖ 0000010r1 ‖ r2 · · · 0 ‖ 0000000m1 ‖ m2 · · ·)e mod N] , which will never decrypt correctly (since the second byte is formatted incorrectly, regardless of the value of r1). 11.15 Consider the RSA-based encryption scheme in which a user encrypts a message m ∈ {0, 1}ℓ with respect to the public key ⟨N, e⟩ by computing m ˆ := H(m)‖m and outputting the ciphertext [m ˆ e mod N]. (Here, let ℓ n H : {0, 1} → {0, 1} and assume ℓ + n < ‖N ‖.) The receiver recovers m ˆ in the usual way and verifies that it has the correct form before outputting the ℓ least-significant bits as m. Prove or disprove that this scheme is CCA-secure if H is modeled as a random oracle. Solution: This scheme is not even CPA-secure, since encryption is deterministic. 11.16 Show a chosen-ciphertext attack on Construction 11.34. Solution: There are many possible answers; we provide one. We only sketch an attack. Given a ciphertext c that encapsulates an unknown key k1 · · · kn , compute c′ := [ce mod N] and request decryption of c′ . The result will be k2 · · · kn k, where k = lsb(c), thus revealing all-butone of the bits of the encapsulated key. 11.17 Let Π = (Gen, Enc, Dec) be a CPA-secure public-key encryption scheme, and let Π′ = (Gen′ , Enc′ , Dec′) be a CCA-secure private-key encryption scheme. Does Construction 11.3-S have indistinguishable encryptions under a chosen-ciphertext attack, if H is modeled as a random oracle? If yes, provide a proof. If not, where does the approach used to prove Theorem 11.38 break down? Solution: Π∗ does not, in general, have indistinguishable encryptions under a chosen-ciphertext attack. The issue is that it might be easy to take a ciphertext c1 encrypted using Π and generate a different ciphertext c′1 that decrypts to the same value as c1 . For example, El Gamal encryption has this property (see Section 11.4.4). If Π has this property, then there is an easy chosen-ciphertext attack on Π∗ : given ciphertext 124 Introduction to Modern Cryptography – 2nd Edition Solutions Manual CONSTRUCTION 11.3-S Let H : {0, 1}n → {0, 1}n be a function. Construct a public-key encryption scheme as follows: • Gen∗ : on input 1n , run Gen(1n) to obtain (pk, sk). Output these as the public and private keys, respectively. • Enc∗ : on input a public key pk and a message m ∈ {0, 1}n , choose a uniform r ∈ {0, 1}n and output the ciphertext ⟨ ⟩ Encpk (r), Enc′H(r) (m) . • Dec∗ : on input a private key sk and a ciphertext ⟨c1 , c2 ⟩, compute r := Decsk (c1) and set k := H(r). Then output Dec′k (c2). C = ⟨c1 , c2 ⟩ generate a ciphertext C ′ = ⟨c′1 , c2 ⟩ (where c′1 decrypts to the same value as c1 with respect to the given public key) and then submit this ciphertext to the decryption oracle. Note that the decryption of C ′ is the same as the decryption of C. The feature of Construction 11.37 that prevents this attack is that textbook RSA encryption is a permutation; that is, every message corresponds to a unique ciphertext. Thus, given a ciphertext c1 encrypted using textbook RSA encryption, there does not exist a different ciphertext c′1 decrypting to the same value. 11.18 Consider the following variant of Construction 11.32: CONSTRUCTION 11.4-S Let GenRSA be as usual, and define a public-key encryption scheme as follows: • Gen: on input 1n , run GenRSA(1n) to obtain (N, e, d). Output the public key pk = ⟨N, e⟩, and the private key sk = ⟨N, d⟩. • Enc: on input a public key pk = ⟨N, e⟩ and a message m ∈ {0, 1}, choose a uniform r ∈ Z∗N . Output the ciphertext ⟨[re mod N], lsb(r) ⊕ m⟩. • Dec: on input a private key sk = ⟨N, d⟩ and a ciphertext ⟨c, b⟩, compute r := [cd mod N] and output lsb(r) ⊕ b. Prove that this scheme is CPA-secure. Discuss its advantages and disadvantages relative to Construction 11.32. Solution: The proof is (conceptually) very similar to the proof of Theorem 11.33. Let Π denote the above variant scheme, and let A be an adversary attacking this scheme. Assume for simplicity (and without loss Public-Key Encryption 125 of generality) that A always outputs m0 = 0 and m1 = 1. Construct an algorithm A′ for experiment RSA-lsb as follows: A′ , given N, e, c, chooses a uniform bit ˆb and computes b′ ← A(N, e, ⟨c, ˆb⟩); it then outputs ˆb ⊕ b′ . Let r be the (unknown) value such that c = [re mod N]. Note that Pr[A′ (N, e, c) = 0 | lsb(r) = 0] 1 = · (Pr[A(N, e, ⟨c, 0⟩) = 0 | lsb(r) = 0] + Pr[A(N, e, ⟨c, 1⟩) = 1 | lsb(r) = 0]) 2 and Pr[A′ (N, e, c) = 1 | lsb(r) = 1] 1 = · (Pr[A(N, e, ⟨c, 0⟩) = 1 | lsb(r) = 1] + Pr[A(N, e, ⟨c, 1⟩) = 0 | lsb(r) = 1]) . 2 Moreover, Pr[A outputs 0 | 0 is encrypted] 1 = · (Pr[A(N, e, ⟨c, 0⟩) = 0 | lsb(r) = 0] + Pr[A(N, e, ⟨c, 1⟩) = 0 | lsb(r) = 1]) , 2 and similarly for Pr[A outputs 1 | 1 is encrypted]. Putting everything together gives Pr[PubKeav A,Π (n) = 1] = Pr[RSA-lsbA′ ,GenRSA (n) = 1], which shows that the scheme is CPA-secure if the RSA problem is hard relative to GenRSA. 11.19 Say three users have RSA public keys ⟨N1 , 3⟩, ⟨N2 , 3⟩, and ⟨N3 , 3⟩ (i.e., they all use e = 3), with N1 < N2 < N3 . Consider the following method for sending the same message m ∈ {0, 1}ℓ to each of these parties: choose a uniform r ← Z∗N1 , and send to everyone the same ciphertext ⟨ 3 ⟩ [r mod N1], [r3 mod N2], [r3 mod N3], H(r) ⊕ m , where H : Z∗N1 → {0, 1}ℓ . Assume ℓ ≫ n. (a) Show that this is not CPA-secure, and an adversary can recover m from the ciphertext even when H is modeled as a random oracle. (b) Show a simple way to fix this and get a CPA-secure method that transmits a ciphertext of length 3ℓ + O(n). (c) Show a better approach that is still CPA-secure but with a ciphertext of length ℓ + O(n). 126 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Solution: (a) The second attack presented in Section 10.4.2 shows how r can be recovered in polynomial time given [r3 mod N1], [r3 mod N2], and [r3 mod N3]. Given r, the message m can be easily recovered. (b) One simple way to fix the scheme is to use an independent value r for each user. I.e., to encrypt the sender would choose random ri ∈ ZNi for i ∈ {1, 2, 3} and then send the ciphertext ⟨ 3 ⟩ [r1 mod N1], [r23 mod N2], [r33 mod N3], . H(r1) ⊕ m, H(r2) ⊕ m, H(r3) ⊕ m (c) A better idea is to use a variant of hybrid encryption: use the approach from part (b) to encrypt a key k, and then use the same key k to encrypt m. In detail: let Π = (Enc, Dec) denote a CPAsecure private-key encryption scheme. Then to encrypt the sender would choose random ri ∈ ZNi for i ∈ {1, 2, 3} as well as random k ∈ {0, 1}n , and then send the ciphertext ⟩ ⟨ 3 [r1 mod N1], [r23 mod N2], [r33 mod N3], H(r1) ⊕ k, H(r2) ⊕ k, H(r3) ⊕ k, . Enck (m) 11.20 Fix an RSA public key ⟨N, e⟩ and assume we have an algorithm A that always correctly computes lsb(x) given [xe mod N]. Write full pseudocode for an algorithm A′ that computes x from [xe mod N]. def Solution: Let γ = [2−1 mod N]. The intuition is that xe · γ e = (xγ)e mod N ; thus, multiplication by γ e can be used to effect a bitwise right-shift (i.e., division by 2), which can in turn be used to learn all the bits of x one-by-one. There are, however, some technicalities. ALGORITHM 11.5-S GetBits Input: ⟨N, e⟩; c ∈ Z∗N ; ℓ Output: the ℓ least significant bits of [c1/e mod N] if ℓ = 1, return A(N, e, c) γ := [2−1 mod N] x0 := A(N, e, c) x′ := GetBits(N, e, [c · γ e mod N], ℓ − 1) if x0 = 0 return x′ ‖x0 else return [2x′ − N mod 2ℓ] When lsb(x) = 0 then [γ ·x mod N] is indeed just a right-shift of x (since x, viewed as an integer, is divisible by 2). But when lsb(x) = 1, then Public-Key Encryption 127 [γ · x mod N] = x+N 2 . (Note that N is odd.) We take this into account in Algorithm 11.5-S, which is described recursively. The algorithm relies on the assumed algorithm A for computing lsb(x). When called with ℓ = ‖N ‖ it returns all the bits of x = [c1/e mod N]. 11.21 Fix an RSA public key ⟨N, e⟩ and define { 0 if 0 < x < N/2 half(x) = 1 if N/2 < x < N Prove that half is a hard-core predicate for the RSA problem. Solution: The key observation is that half(x) = 0 ⇔ lsb([2x mod N]) = 0. This is because when x < N/2 then [2x mod N] = 2x (there is no overflow), which is even. On the other hand, when x > N/2 then [2x mod N] = 2x − N , which is odd. Turning this around, given an algorithm A for computing half(x) from N, e, [xe mod N], we can compute lsb(x) from N, e, c = [xe mod N] by calling A(N, e, [c/2e mod N]). Chapter 12 Digital Signature Schemes – Solutions 12.1 Show that Construction 4.7 for constructing a variable-length MAC from any fixed-length MAC can also be used (with appropriate modifications) to construct a signature scheme for arbitrary-length messages from any signature scheme for messages of fixed length ℓ(n) ≥ n. Solution: The proof that Construction 4.7 works also in the context of signatures follows along exactly the same lines as the proof of Theorem 4.8. As long as ℓ(n) = ω(log n) (see Appendix A.2 for a definition of this notation), the construction works. To see this, recall that the construction needs to compute a signature on "messages" of the form r‖len‖i‖mi , where r is a uniform string of some length; len denotes the message length (encoded in binary); i denotes the block number (encoded in binary); and mi is the i block of the message. Using 1-bit blocks (so |mi | = 1), we need |len| to be ω(log n) so that messages of any polynomial length can be handled (at least asymptotically). It then suffices for |i| to have length ω(log n) as well. For the security proof, we require |r| = ω(log n) so that a "repeat" does not occur too often (cf. the proof of Claim 4.7). Altogether, then, ℓ(n) = ω(log n) is sufficient. 12.2 Prove that the existence of a one-time-secure signature scheme for 1-bit messages implies the existence of one-way functions. Solution: Let (Gen, Sign, Vrfy) be a one-time signature scheme for 1-bit messages. Since Gen runs in polynomial time, there exists a polynomial p such that the number of random bits the algorithm uses on input 1n is at most p(n). For simplicity, assume that Gen always uses exactly p(n) random bits on input 1n , and further than p(n) is strictly increasing. Define the following function f : on input a string x, compute n such that p(n) ≤ |x| ≤ p(n + 1). Then compute (pk, sk) := Gen(1n ; x) and output pk. A proof that f as defined above is one-way follows along the same lines as the proof of Theorem 8.74. The only difference here is to note that inverting f on a given pk allows computation of an associated private 129 130 Introduction to Modern Cryptography – 2nd Edition Solutions Manual key sk ′ (which need not be equal to sk), which then enables signature forgery with respect to pk. 12.3 In Section 12.4 we showed an attack on the plain RSA signature scheme in which an attacker forges a signature on an arbitrary message using two signing queries. Show how an attacker can forge a signature on an arbitrary message using a single signing query. Solution: Let (N, e) be the public key and let x ∈ ZN be the message that the attacker A wishes to obtain a signature for. Then, A chooses an arbitrary r ∈ ZN and computes s = re mod N . Then, A asks for a signature on x · s mod N and receives back some σ. Finally, A outputs the pair (x, σ ˆ) where σ ˆ = σr mod N . In order to see why this is correct, observe that σ ˆe = σe re = ((x·s)d)e re = x·s re = x·r e re = x mod N , as required. 12.4 Assume the RSA problem is hard. Show that the plain RSA signature scheme satisfies the following weak definition of security: an attacker is given the public key ⟨N, e⟩ and a uniform message m ∈ Z∗N . The adversary succeeds if it can output a valid signature on m without making any signing queries. Solution: A formal proof follows easily from the fact that computing a signature on a random message m is equivalent to solving the RSA problem on a random instance. 12.5 Another approach (besides hashing) that has been tried to construct secure RSA-based signatures is to encode the message before applying the RSA permutation. Here the signer fixes a public encoding function enc : {0, 1}ℓ → Z∗N as part of its public key, and the signature on a message m is σ := [enc(m)d mod N]. (a) How is verification performed in encoded RSA? (b) Discuss why appropriate choice of encoding function for ℓ ≪ ‖N ‖ prevents the "no-message attack" described in Section 12.4.1. (c) Show that encoded RSA is insecure if enc(m) = 0κ/10 ‖m‖0κ/10 def def (where κ = ‖N ‖, ℓ = |m| = 4κ/5, and m is not the all-0 message). Assume e = 3. (d) Show that encoded RSA is insecure for enc(m) = 0‖m‖0‖m (where def ℓ = |m| = (‖N ‖ − 1)/2 and m is not the all-0 message). Assume e = 3. (e) Solve parts (c) and (d) for arbitrary e. Solution: (a) To verify a signature σ on a message m with respect to a public key pk = ⟨N, e⟩, simply check whether ? σ e = enc(m) mod N. Digital Signature Schemes 131 (b) The "no-message attack" from Section 12.3.1 would work by choosing a uniform σ ∈ Z∗N and then trying to find an m such that enc(m) = σ e mod N . If enc is such that a uniform element of Z∗N lies in the range of enc with only negligible probability, then this attack will not work since [σ e mod N] is not even in the range of enc except with negligible probability. We remark that the encoding schemes used in parts (c) and (d) both have the above property. (c) The key observation is that enc(m) = m · 2κ/10 mod N , as long as m can be expressed as an ℓ-bit binary integer. An attack on the scheme is as follows: Set m := 0 · · · 01000 = 2e and request a signature σ on m. Then output the forged signature [σ · 2−1 mod N] on the message m′ = 0 · · · 01 = 1. This works since ( )e σ · 2−1 = enc(m) · 2−e = 2e 2κ/10 2−e = 2κ/10 = enc(m′) mod N. (d) Observe that enc(m) = m · (2ℓ+1 + 1) mod N as long as m can be expressed as an ℓ-bit binary integer. The same attack as in the previous part works here as well. (e) Both parts use an encoding of the form enc(m) = m · α, so we show one attack that applies to both. Request a signature σ4 on the message 0 · · · 0100 = 4 and a signature σ2 on the message 0 · · · 010 = 2. Then output the forgery σ := [σ42 /σ2 mod N] on the message 0 · · · 01000 = 8. This is a valid forgery since ( )e σ e = σ42 /σ2 = enc(4)2 /enc(2) 2 (4 · α) 2·α = 8 · α = enc(8) mod N. = 12.6 Consider a variant of the Fiat–Shamir transform in which the signature is (I, s) rather than (r, s) and verification is changed in the natural way. Show that if the underlying identification scheme is secure, then the resulting signature scheme is secure here as well. Solution: This follows immediately from the fact that, given only the public key, it is possible to convert a signature of the first type to a signature of the second type, and vice versa. 12.7 Consider a variant of DSA in which the message space is Zq and H is omitted. (So the second component of the signature is now s := [k −1 · (m + xr) mod q].) Show that this variant is not secure. Solution: The key observation is that k can be set implicitly and then m can be chosen to make things cancel appropriately. The easiest example 132 Introduction to Modern Cryptography – 2nd Edition Solutions Manual is to set k implicitly equal to x, by using y in place of g k in the signing calculation. I.e., compute r := F (y) (so k = x), and then observe that s := [k −1 · (m + xr) mod q] = [x−1 m + r mod q]. Thus, if we set m := 0 then s can be computed without knowing x. More generally, we can compute r := F (g α y β) so k = α + βx. Then s := [(α + βx)−1 · (m + xr) mod q], and we see that by setting m = rα/β it is possible to compute s without knowing x. 12.8 Let f be a one-way permutation. Consider the following signature scheme for messages in the set {1, . . . , n}: • To generate keys, choose uniform x ∈ {0, 1}n and set y := f (n) (x) def (where f (i) (·) refers to i-fold iteration of f , and f (0) (x) = x). The public key is y and the private key is x. • To sign message i ∈ {1, . . . , n}, output f (n−i) (x). • To verify signature σ on message i with respect to public key y, ? check whether y = f (i) (σ). (a) Show that the above is not a one-time-secure signature scheme. Given a signature on a message i, for what messages j can an adversary output a forgery? (b) Prove that no ppt adversary given a signature on i can output a forgery on any message j > i except with negligible probability. (c) Suggest how to modify the scheme so as to obtain a one-time-secure signature scheme. Solution: (a) Here is an attack. Request a signature on the message n. This gives the secret key x = f n−n (x), after which a signature on any other message can be computed. More generally, given a signature on a message i it is possible to forge a signature for any message j with j < i. (In detail, if σi is a def signature on i, then σj = f i−j (σi) is a signature on j. Since i > j and f is known, computation of σj can be done efficiently.) (b) We first give a formal security definition. Consider the following experiment for an adversary A and security parameter n (the experiment is tailored for the scheme proposed in this exercise): Digital Signature Schemes 133 The modified signature experiment Sig-forge′A (n): i. Choose x ← {0, 1}n and set y := f n (x). ii. Adversary A is given y. It then outputs any i ∈ {1, . . . , n} and is given in return the signature σi = f n−i (x) on i. iii. A outputs (j, σj) with j > i. iv. The output of the experiment is defined to be 1 if and only if f j (σj) = y. DEFINITION The signature scheme is weakly secure if for all probabilistic polynomial-time adversaries A, there exists a negligible function negl such that Pr[Sig-forge′A (n) = 1] ≤ negl(n). We now show that if f is a one-way permutation then the scheme is weakly secure. Let A be any ppt adversary. Consider the following algorithm trying to invert f on a given point: Algorithm D: The algorithm is given z and its goal is to compute f −1 (z). • Choose i∗ ← {1, . . . , n}. ∗ • Set y = f i (z). • Run A on the public key y. When A requests a signature on i, abort unless i = i∗ . If i = i∗ then give the adversary the signature σi = z. ∗ • When A outputs (j, σj), output f j−i −1 (σj). Let us analyze D when its input z satisfies z = f (x) for uniform x ∈ {0, 1}n . Note first that, since f is a permutation, the public key y is uniform in {0, 1}n just as in the real signature scheme. Furthermore, the probability that i∗ = i is exactly 1/n because i∗ is chosen uniformly from {1, . . . , n}. Conditioned on this being the case, the signature σi given to the adversary is correct. Finally, if A outputs a valid forgery, then the output of D is indeed the inverse of z. Putting everything together, we have: Pr[InvertD,f (n) = 1] ≥ 1 · Pr[Sig-forge′A (n) = 1]. n If f is one-way, then this implies the existence of a negligible function negl such that Pr[Sig-forge′A (n) = 1] < negl(n). (c) Consider the following modified scheme, which signs messages in {1, . . . , n − 1}: • To generate keys, choose uniform x, x′ ← {0, 1}n and set y := f n (x) and y ′ := f n (x′). The public key is ⟨y, y ′ ⟩ and the private key is ⟨x, x′ ⟩. 134 Introduction to Modern Cryptography – 2nd Edition Solutions Manual • To sign message i ∈ {1, . . . , n}, output σ = f n−i (x) and σ ′ = f i (x′). • To verify signature ⟨σ, σ ′ ⟩ on message i with respect to public ? ? key ⟨y, y ′ ⟩, check whether y = f i (σ) and y ′ = f n−i (σ ′). Note that a signature on i in the modified scheme consists of two signatures in the original scheme: one on i, and one on n−i. Given the result from part (b), one-time security of this modified scheme follows easily. 12.9 A strong one-time-secure signature scheme satisfies the following (informally): given a signature σ ′ on a message m′ , it is infeasible to output (m, σ) ≠ (m′ , σ ′) for which σ is a valid signature on m (note that m = m′ is allowed). (a) Give a formal definition of strong one-time-secure signatures. (b) Assuming the existence of one-way functions, show a one-way function for which Lamport's scheme is not a strong one-time-secure signature scheme. (c) Construct a strong one-time-secure signature scheme based on any assumption used in this book. Solution: (a) Let Π = (Gen, Sign, Vrfy) be a signature scheme, and consider the following experiment for an adversary A and parameter n: The strong one-time signature experiment -strong (n): Sig-forge1A,Π i. Gen(1n) is run to obtain keys (pk, sk). ii. Adversary A is given pk and asks a single query m′ to oracle Signsk (·). Let σ ′ denote the signature that was returned. A then outputs (m, σ) where (m, σ) ≠ (m′ , σ ′). iii. The output of the experiment is defined to be 1 if and only if Vrfypk (m, σ) = 1. DEFINITION A signature scheme Π = (Gen, Sign, Vrfy) is a strong one-time signature scheme, if for all probabilistic polynomialtime adversaries A, there exists a negligible function negl such that: -strong (n) = 1] ≤ negl(n). Pr[Sig-forge1A,Π (b) Let f be a one-way function and define f ′ as follows: f ′ (x‖b) = f (x) (where we let b denote the last bit of the input to f ′). It is not Digital Signature Schemes 135 hard to see that f ′ is a one-way function, but given an inverse of any point y it is easy to come up with a different inverse of y. (Namely, given x′ = x‖b such that f ′ (x′) = y we know that x′′ = x‖¯b also satisfies f ′ (x′′) = y.) When f ′ is used in Lamport's scheme, the resulting signature scheme is clearly not a strong one-time signature scheme. E.g., take ℓ = 1 and note that if x′ = x‖b is a signature on the message 0, then so is x′′ = x‖¯b. (c) Take f to be a collision-resistant hash function mapping, say, 2nbit strings to n-bit outputs. (See the next exercise for a proof that such a collision-resistant hash function is one-way.) Formally, as part of key generation the signer generates a uniform key s for a collision-resistant hash function and includes s as part of the public key; the rest is exactly as in Construction 12.7 except that H s is used in place of f . Assuming the result of the following exercise, this scheme is certainly a one-time signature scheme. But any two signatures σ, σ ′ on the same message m yield a collision in the hash function. A formal proof that this scheme is a strong one-time signature scheme is immediate. 12.10 Consider the Lamport signature scheme. Describe an adversary who obtains signatures on two messages of its choice and can then forge signatures on any message it likes. Solution: The attack is simple. Consider the scheme defined for messages of length ℓ = ℓ(n). The adversary can request signatures on the two messages 0ℓ and 1ℓ . The values contained in the two signatures together yield the entire private key, at which point a signature on any message can be forged. 12.11 The Lamport scheme uses 2ℓ values in the public key to sign messages of length ℓ. Consider the variant in which the private key contains 2ℓ values x1 , . . . , x2ℓ and the public key contains the values y1 , . . . , y2ℓ with ′ yi := f (xi). A message m ∈ {0, 1}ℓ is mapped in a one-to-one fashion to a subset Sm ⊂ {1, . . . , 2ℓ} of size ℓ. To sign m, the signer reveals {xi }i∈Sm . Prove that this gives a one-time-secure signature scheme. What is the maximum message length ℓ′ that this scheme supports? Solution: A proof that this gives a one-time signature scheme is very similar to the proof of Theorem 12.8. Let Π denote the scheme, and let A be a probabilistic polynomial-time adversary. Consider the following ppt algorithm I attempting to invert the one-way function f : Algorithm I: The algorithm is given y and 1n as input. 136 Introduction to Modern Cryptography – 2nd Edition Solutions Manual • Choose i∗ ← {1, . . . , 2ℓ}, and set yi∗ := y. • For i ∈ {1, . . . , 2ℓ} \ {i∗ }, do: – Choose xi ← {0, 1}n . – Set yi := f (xi). • Set pk = (y1 , . . . , y2ℓ) and run A(pk). • When A requests a signature of the message m′ , map m′ to a subset Sm′ of size ℓ. If i∗ ∈ Sm′ , then abort. Otherwise, give {xi }i∈Sm′ to A. • When A outputs (m, σ = (xi1 , . . . , xiℓ)), map m to a subset Sm of size ℓ. If i∗ is the first index for which i∗ ∈ Sm but i∗ ∉ Sm′ , then output xi∗ . Otherwise, abort. In an execution of A, let m′ denote the message whose signature is requested by A, and let m ≠ m′ denote the message whose signature is (purportedly) forged by A. Let Sm′ and Sm be defined in the natural way, and let j ∈ {1, . . . , 2ℓ} denote the first index such that j ∈ Sm but j ∉ Sm′ . (Note that such j must always exist.) In the execution of I, above, observe that the public key given to A is distributed identically to the public key in a real execution of Π. Furthermore, if I correctly guesses i∗ = j, then the signature given by I to A is correct. Finally, conditioned on i∗ = j, whenever A outputs a valid forgery it is the case that I correctly inverts the given value y. Since i∗ = j with probability exactly 1/2ℓ, we have Pr[InvertI,f (n) = 1] = 1 -time (n) = 1]. · Pr[Sig-forge1A,Π 2ℓ Since f is one-way, this implies the existence of a negligible function -time (n) = 1] < negl(n). negl such that Pr[Sig-forge1A,Π ( ) The number of subsets of size ℓ is 2ℓ ℓ , and so this is the number of messages that(can be signed. So the maximum message length is given) by ℓ′ = ⌊log2 2ℓ ⌋. ℓ 12.12 At the end of Section 12.6.2, we show how a pseudorandom function can be used to make Construction 12.12 stateless. Does a similar approach work for the chain-based scheme described in Section 12.6.1? If so, sketch a construction and proof. If not, explain why and modify the scheme to obtain a stateless variant. Solution: A similar approach cannot be directly applied to the chainbased scheme, since that scheme required the signer's state to include the entire history of messages that have been signed so far. We describe a way to adapt the chain-based scheme so that the message history does not need to be stored as part of the signer's state; this variant can be made stateless following the approach used in Section 12.6.2. Digital Signature Schemes 137 On a high level, the modification will be to use each public key pki to ′ ′ sign the concatenation of two public keys pki+1 ‖pki+1 , and to use pki+1 to sign the message mi . (Looking at Figure 12.2, imagine replacing each ′ message mi with a public key pki+1 along with an additional arrow from ′ pki+1 to mi .) We now give the details. In the variant we describe, the public key again consists of just a single public key pk1 generated using Gen and the private key contains the associated private key sk1 . To sign the first message m1 , the signer first generates two new key-pairs (pk2 , sk2), (pk2′ , sk2′) using Gen, and then signs both pk2′ and pk2 using sk1 to obtain σ1 ← Signsk1 (pk2′ ‖pk2). Finally, the signer signs m1 using sk2′ to obtain σ1′ ← Signsk2′ (m1). The signature that is output includes pk2′ , pk2 , σ1 , and σ1′ and the signer adds (pk2′ , pk2 , sk2 , σ1) to its current state. We stress that neither m1 nor σ1′ need to be stored. In general, when it comes time to sign the ith message the signer will ′ , pkj+1 , skj+1 , σj)}i−1 have stored {(pkj+1 j=1 as part of its state. To sign the ith message mi , the signer generates two new key-pairs (pki+1 , ski+1), ′ ′ ′ and pki+1 using ski to obtain) using Gen; signs pki+1 , ski+1 (pki+1 ′ ′ a signature σi ← Signski (pki+1 ‖pki+1); and signs mi using ski+1 to ′ obtain σi ← Signski+1 (mi). The actual signature that is output in′ ′ cludes pki+1 , pki+1 , σi , σi′ , and also the values {pkj+1 , pkj+1 , σj }i−1 j=1 . The signer then adds (pki+1 , pki+1 , ski+1 , σi) to its state. Verification is done in the expected way. As in the case of the original chain-based scheme, security follows on an intuitive level from the fact that every key-pair generated by the signer is used to sign only a single "message". 12.13 Prove Theorem 12.22. Solution: Let Π∗ denote Construction 12.20 and let Π denote the stateless variant, instantiated with a pseudorandom function F as discussed in the book. We assume that the signing algorithm of the underlying one-time signature scheme used in Construction 12.20 is deterministic. Fix a ppt adversary A. We now construct an algorithm D that is given access to an oracle O(·) and must decide whether O(·) is a function chosen truly at random, or whether O(·) = Fk (·) for a uniform key k. Algorithm D: The algorithm is given 1n and access to an oracle O(·). • Implement Construction 12.20 for A. In the course of this experiment, every time randomness rw is needed to generate (pkw , skw) the algorithm obtains rw by querying O(w). 138 Introduction to Modern Cryptography – 2nd Edition Solutions Manual • Eventually, A outputs a message/signature pair (m, σ). If this is a valid forgery then A outputs 1; otherwise, it outputs 0. It is easy to see that if O(·) is a function chosen at random, then D exactly implements scheme Π∗ for A; on the other hand, if O(·) = Fk (·) for a uniform key k then D implements Π for A. Therefore, Pr[DFk (·) (1n) = 1] − Pr[Df (·) (1n) = 1] = Pr[Sig-forgeA,Π (n) = 1] − Pr[Sig-forgeA,Π∗ (n) = 1] By Theorem 12.21 says that there exists a negligible function negl for which Pr[Sig-forgeA,Π∗ (n) = 1] ≤ negl(n). The fact that F is a pseudorandom function means that there is a negligible function negl′ such that Pr[DFk (·) (1n) = 1] − Pr[Df (·) (1n) = 1] ≤ negl′ (n). Thus, Pr[Sig-forgeA,Π (n) = 1] ≤ negl(n) + negl′ (n). Since A was arbitrary, this proves the theorem. 12.14 Assume revocation of certificates is handled in the following way: when a user Bob claims that the private key corresponding to his public key pkB has been stolen, the user sends to the CA a statement of this fact signed with respect to pkB . Upon receiving such a signed message, the CA revokes the appropriate certificate. Explain why it is not necessary for the CA to check Bob's identity in this case. In particular, explain why it is of no concern that an adversary who has stolen Bob's private key can forge signatures with respect to pkB . Solution: Say the CA receives a message, signed with respect to pkB , requesting revocation of B's key. There are two cases: either B signed this message or not. If B signed the message, then the legitimate user is requesting revocation of the key and so it is ok for the CA to revoke it. If B did not sign the message, then B's key must have been compromised (or the scheme has been broken in some other way) in which case B would want its key to be revoked, anyway! Chapter 13 Advanced Topics in Public-Key Encryption – Solutions 13.1 Construct and prove CPA-security for a KEM based on any trapdoor permutation by suitably generalizing Construction 11.34. Solution: See Construction 13.1-S. A proof of security is similar to that of Theorem 11.35. CONSTRUCTION 13.1-S d f, Inv) be a family of trapdoor permutations with hardb = (Gen, Let Π core predicate hc. Define a public-key encryption scheme as follows: d n) to obtain (I, td). Output the • Gen: on input 1n , run Gen(1 public key I and the private key td. • Encaps: on input pk = I and 1n , choose uniform c1 ∈ DI . Then for i = 1, . . . , n do: (a) Compute ki := hcI (ci). (b) Compute ci+1 := fI (ci). Output the ciphertext cn+1 and the key k = k1 · · · kn . • Decaps: on input sk = td, a ciphertext cn+1 , and 1n , for i = n, . . . , 1 do: (a) Compute ci := InvI (ci+1). (b) Compute ki := hcI (ci). Output the key k = k1 · · · kn . A KEM based on any trapdoor permutation. 13.2 Show that the isomorphism of Proposition 13.6 can be efficiently inverted when the factorization of N is known. Solution: We are given y ∈ Z∗N 2 and want to find a ∈ ZN and b ∈ Z∗N such that y = (1 + N)a · bN mod N 2 . 139 140 Introduction to Modern Cryptography – 2nd Edition Solutions Manual The decryption algorithm of the Paillier scheme allows recovery of a. To recover b, we compute [ ] −1 b := [y mod N][N mod ϕ(N)] mod N (recall that gcd(ϕ(N), N) = 1 and so [N −1 mod ϕ(N)] is well-defined). Let us see why this works. We claim first that [y mod N] = [bN mod N]. To see this, observe that (1 + N)a · bN = y + r · N 2 for some integer r. (Note that this is an equation over the integers.) So [ ] [y mod N] = (1 + N)a · bN − rN 2 mod N = [1 · bN mod N]. Given this, we have [ ] [( )[N −1 mod ϕ(N)] [N −1 mod ϕ(N)] N [y mod N] mod N = b mod N [ ] −1 = b[N ·N mod ϕ(N)] = [b mod N] = b. 13.3 Let Φ(N 2) denote the set {(a, 1) | a ∈ ZN } ⊂ Z∗N 2 . Show that it is not hard to decide whether a given element y ∈ Z∗N 2 is in Φ(N 2). Solution: Using Proposition 13.6(2), we have that Φ(N 2) = {[1 + aN mod N 2] | a ∈ ZN }. It is easy to decide whether y ∈ Φ(N 2) by checking whether the integer y − 1 is divisible by N . 13.4 Let G be an abelian group. Show that the set of quadratic residues in G forms a subgroup. Solution: Let Q = {x2 | x ∈ G} denote the set of quadratic residues. We show that Q is a subgroup. • Let a, b ∈ Q. So there exist x, y ∈ G such that a = x2 and b = y 2 . Then ab = x2 · y 2 = (xy)2 , and so ab ∈ Q. (Note we use here the fact that G is abelian.) • Let 1 denote the identity in G. Then 12 = 1 and so 1 ∈ Q. • Let a ∈ Q. So there exists x ∈ G such that a = x2 . Then ( )−1 (−1)2 , = x a−1 = x2 and so a−1 ∈ Q. Advanced Topics in Public-Key Encryption 141 • Associativity in Q is inherited from G. 13.5 This question concerns the quadratic residues in the additive group ZN . (An element y ∈ ZN is a quadratic residue if and only if there exists an x ∈ ZN with 2x = y mod N .) (a) What are the quadratic residues in Zp for p an odd prime? (b) Let N = pq be a product of two odd primes p and q. What are the quadratic residues in ZN ? (c) Let N be an even integer. What are the quadratic residues in ZN ? Solution: (a) All elements are quadratic residues. To see this, note that for any a ∈ Zp , the equation a = 2x mod p has the solution x = [2−1 · a mod p]. (This follows since 2 is invertible modulo p.) (b) As in the previous part, all elements are quadratic residues. (c) Only even integers are quadratic residues in ZN . If a ∈ ZN is even, then the equation a = 2x mod N clearly has the solution x = a/2 (it also has the solution x = (N + a)/2). On the other hand, if a is odd then a = 2x mod N cannot have a solution since, if it did, then there would exist an integer b with a = 2x + bN in contradiction to the fact that a is odd. 13.6 Let N = pq with p, q distinct, odd primes. Show a ppt algorithm for choosing a uniform element of QN R+1 N when the factorization of N is known. (Your algorithm can have failure probability negligible in ‖N ‖.) Solution: We can choose a uniform quadratic non-residue xp ∈ QN Rp by repeatedly choosing uniform elements from Z∗p until we find one that is not a quadratic residue; since half the elements of Z∗p are not quadratic residues, if we choose n uniform elements from Z∗p this procedure fails with only negligible probability. A uniform quadratic non-residue xq ∈ QN Rq can be chosen similarly. We then use the bijection provided by the Chinese remainder theorem to compute x ∈ Z∗N with (xp , xq) ↔ x. 13.7 Let N = pq with p, q distinct, odd primes. Prove that if x ∈ QRN then −1 [x−1 mod N] ∈ QRN , and if x ∈ QN R+1 N then [x−1 mod N] ∈ QN R+1 N . Solution: If x ∈ QRN then there exists a y ∈ Z∗N such that x = y 2 mod N . So ( )−1 (−1)2 mod N , = y x−1 = y 2 and [x−1 mod N] is a quadratic residue. Let x ↔ (xp , xq). If x ∈ QN R+1 N , then this implies that xp ∈ QN Rp and xq ∈ QN Rq . This means that [x−1 mod p] ∈ QN Rp and [x−1 q mod (−1)−1p q] ∈ QN Rq . (If not, then [xp mod p] ∈ QRp , a contradiction.) 142 Introduction to Modern Cryptography – 2nd Edition Solutions Manual ( ) −1 Since [x−1 mod N] ↔ [x−1 p mod p], [xq mod q] , this means that we have [x−1 mod N] ∈ QN R+1 N . 13.8 Let N = pq with p, q distinct, odd primes, and fix z ∈ QN R+1 N . Show that choosing uniform x ∈ QRN and setting y := [z · x mod N] gives a y that is uniformly distributed in QN R+1 ˆ ∈ QN R+1 N . That is, for any y N Pr[z · x = y ˆ mod N] = 1/|QN R+1 N |, where the probability is taken over uniform choice of x ∈ QRN . Solution: zx = y ˆ mod N iff x = z −1 y ˆ mod N . By the previous exercise, we know that [z −1 mod N] ∈ QN R+1 N . Using Corollary 13.25(2), we see that [z −1 y ˆ mod N] ∈ QRN . So Pr[zx = y ˆ mod N] = Pr[x = z −1 y ˆ mod N] = 1/|QRN |. Since |QRN | = |QN R+1 N |, this proves the desired result. 13.9 Let N be the product of 5 distinct odd primes. If y ∈ Z∗N is a quadratic residue, how many solutions are there to the equation x2 = y mod N ? ∏5 Solution: Let N = i=1 pi , with the pi distinct odd primes. Applying the Chinese remainder theorem several times, we see there is an isomorphism x ↔ (x1 , x2 , x3 , x4 , x5) between elements x ∈ Z∗N and elements (x1 , x2 , x3 , x4 , x5) ∈ ×5i=1 Z∗pi . Using arguments as in this chapter, y ↔ (y1 , y2 , y3 , y4 , y5) is a quadratic residue modulo N iff each yi is a quadratic residue modulo pi , and x ↔ (x1 , x2 , x3 , x4 , x5) is a square root of y modulo N iff each xi is a square root of yi modulo pi . Since each yi has two square roots modulo pi , there are 25 square roots of y modulo N . 13.10 Show that the Goldwasser-Micali encryption scheme is homomorphic if the message space {0, 1} is viewed as the group Z2 . Solution: In Goldwasser-Micali encryption, a ciphertext corresponding to the bit m is of the form [z m · x2 mod N] for some x ∈ Z∗N . So if c1 , c2 are encryptions of m1 , m2 , respectively, we have def c = c1 · c2 = z m1 · x21 · z m2 · x22 = z m1 +m2 · (x1 x2)2 mod N. If either m1 or m2 is 0, then c is clearly a ciphertext corresponding to the bit [m1 + m2 mod 2]. If m1 = m2 = 1, then c = z 2 · (x1 x2)2 = z 0 · (z · x1 x2)2 mod N, and so here also c is a ciphertext corresponding to [m1 + m2 mod 2]. Advanced Topics in Public-Key Encryption 143 13.11 Consider the following variation of the Goldwasser-Micali encryption scheme: GenModulus(1n) is run to obtain (N, p, q) where N = pq and p = q = 3 mod 4. (I.e., N is a Blum integer.) The public key is N and the secret key is ⟨p, q⟩. To encrypt m ∈ {0, 1}, the sender chooses uniform x ∈ ZN and computes the ciphertext c := [(−1)m · x2 mod N]. (a) Prove that for N of the stated form, [−1 mod N] ∈ QN R+1 N . (b) Prove that the scheme described has indistinguishable encryptions under a chosen-plaintext attack if deciding quadratic residuosity is hard relative to GenModulus. Solution: (a) Looking at Algorithm 13.18, we see that [−1 mod p] is a quadratic non-residue when p = 3 mod 4. This holds since p = 4s + 3 for some integer s, and so (−1) p−1 2 = (−1)2s+1 = −1 mod p. Similarly, [−1 mod q] is a quadratic non-residue. It follows that [−1 mod N] ∈ QN R+1 N . (b) Let Π denote the modified scheme. We prove that Π has indistinguishable encryptions in the presence of an eavesdropper. Let A be a probabilistic polynomial-time adversary, and define def ε(n) = Pr[PubKeav A,Π (n) = 1]. Consider the following ppt adversary D that attempts to solve the quadratic residuosity problem relative to GenModulus: Algorithm D: The algorithm is given N and z as input and its goal is to determine if z ∈ QRN or z ∈ QN R+1 N . • Set pk = N and run A(pk) to obtain m0 , m1 . Assume without loss of generality that m0 = 0 and m1 = 1. • Set c := z. Give the ciphertext c to A and obtain a bit b′ . Output b′ . Observe that Pr[D(N, qr) = 0] = Pr[A outputs 0 | c is an encryption of 0], where qr denotes a uniform quadratic residue. This follows because an encryption of 0 in Π is just a uniform quadratic residue. Similarly, Pr[D(N, qnr) = 0] = Pr[A outputs 0 | c is an encryption of 1], 144 Introduction to Modern Cryptography – 2nd Edition Solutions Manual where qnr denotes a uniform element of QN R+1 N . This holds since an encryption of 1 in Π is just a uniform element in QN R+1 N . So Pr[D(N, qr) = 1] − Pr[D(N, qnr) = 1] = Pr[D(N, qr) = 0] − Pr[D(N, qnr) = 0] = Pr[A outputs 0 | c is an encryption of 0] − Pr[A outputs 0 | c is an encryption of 1] = Pr[A outputs 0 | c is an encryption of 0 + Pr[A outputs 1 | c is an encryption of 1] − 1 1 = 2 · ε(n) − . 2 So if deciding quadratic residuosity is hard relative to GenModulus, there must be a negligible function negl with ε(n) ≤ 12 + negl(n). 13.12 Assume deciding quadratic residuosity is hard for GenModulus. Show that this implies the hardness of distinguishing a uniform element of QRN from a uniform element of JN+1 . Solution: Fix a probabilistic polynomial-time algorithm D, and define def ε(n) = |Pr[D(N, qr) = 1] − Pr[D(N, J) = 1]| , where the probabilities are taken over N generated by GenModulus and, in the first case, qr chosen uniformly from QRN and, in the second case, J chosen uniformly from JN+1 . We know that with probability 1/2 we have J ∈ QRN (in which case J is uniformly distributed over QRN) and with probability 1/2 we have J ∈ QN R+1 N (in which case J is uniformly distributed over QN R+1). So, N P r[D(N, J) = 1] = 1 1 · Pr[D(N, qr) = 1] + · Pr[D(N, qnr) = 1], 2 2 where qnr is chosen uniformly from QN R+1 N . Thus, ε(n) = 1 · |Pr[D(N, qr) = 1] − Pr[D(N, qnr) = 1]| . 2 So if deciding quadratic residuosity is hard relative to GenModulus, there must exist a negligible function negl with ε(n) ≤ negl(n). 13.13 Show that plain RSA encryption of a message m leaks JN (m). Advanced Topics in Public-Key Encryption 145 Solution: We prove that Jp (me) = Jp (m) when p is prime and e is relatively prime to p − 1. This implies that JN (me) = JN (m). Since JN (x) can be computed efficiently for any x, it follows that plain RSA encryption of m leaks JN (m). To see that Jp (me) = Jp (m), note first that if m = x2 mod p is a quadratic residue modulo p, then so is me = (xe)2 mod p. Moreover, if me = y 2 mod p is a quadratic residue modulo p, then so is m = (me)d = (y d)2 mod p, where d is such that ed = 1 mod p − 1. 13.14 Consider the following variation of the Goldwasser-Micali encryption scheme: GenModulus(1n) is run to obtain (N, p, q). The public key is N and the secret key is ⟨p, q⟩. To encrypt a 0, the sender chooses n uniform elements c1 , . . . , cn ← QRN . To encrypt a 1, the sender chooses n uniform elements c1 , . . . , cn ← JN+1 . In each case, the resulting ciphertext is c∗ = ⟨c1 , . . . , cn ⟩. (a) Show how the sender can generate a uniform element of JN+1 in polynomial time. (b) Suggest a way for the receiver to decrypt efficiently, though with error probability negligible in n. (c) Prove that if deciding quadratic residuosity is hard relative to GenModulus, then this scheme is CPA-secure. Solution: (a) A uniform element of JN+1 can be generated in polynomial time by repeatedly sampling elements from Z∗N and outputting the first such element with Jacobi symbol +1. This works because (1) it is possible to compute the Jacobi symbol in polynomial time, and (2) half the elements of Z∗N are in JN+1 . (b) Given a ciphertext ⟨c1 , . . . , cn ⟩, the receiver can determine whether each ci is a quadratic residue or not. If every ci is a quadratic residue, the receiver outputs 0; otherwise, the receiver outputs 1. When ⟨c1 , . . . , cn ⟩ is an encryption of 0, then ci ∈ QRN for all i and so the receiver always outputs the correct result. When ⟨c1 , . . . , cn ⟩ is an encryption of 1, then for every i we have ci ∈ QRN with probability 1/2 and so the probability that ci ∈ QRN for all i is 2−n . Thus, the receiver errs in this case only with probability 2−n . (c) Let Π denote the encryption scheme above. We prove that Π has indistinguishable encryptions in the presence of an eavesdropper; as usual, this implies that it is CPA-secure. Let A be a probabilistic polynomial-time adversary, and define def ε(n) = Pr[PubKeav A,Π (n) = 1]. 146 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Consider the following ppt adversary D that attempts to distinguish uniform quadratic residues from uniform elements of QN R+1 N : Algorithm D: The algorithm is given N and z as input and its goal is to determine if z ∈ QRN or z ∈ QN R+1 N . • Set pk = N and run A(pk) to obtain m0 , m1 . Assume without loss of generality that m0 = 0 and m1 = 1. • For i = 1 to n, choose uniform xi ∈ QRN and uniform bi ∈ {0, 1}, and set ci := [z bi · xi mod N]. • Give the ciphertext c = ⟨c1 , . . . , cn ⟩ to A and obtain an output bit b′ . Output 1 − b′ . The thing to notice about the algorithm above is the following: if z ∈ QRN , then each ci is a uniform element of QRN and hence c corresponds to a random encryption of 0. This follows since for any y ∈ QRN we have 1 1 · Pr[z 0 · xi = y] + · Pr[z 1 · xi = y] 2 2 1 1 1 1 = · + · Pr[xi = y/z] = , 2 |QRN | 2 |QRN | Pr[z bi · xi = y] = where xi is chosen uniformly from QRN in the above. On the other +1 hand, when z ∈ QN R+1 N then each ci is a uniform element of JN and so c corresponds to a random encryption of 1. This is true because for any y ∈ JN+1 we have Pr[z bi · xi = y] = 1 1 · Pr[xi = y] + · Pr[xi = y/z], 2 2 while { Pr[xi = y] = 1 |QRN | 0 y ∈ QRN y ∈ QN R+1 N and (cf. Exercise 13.7) { Pr[xi = y/z] = 0 1 |QRN | y ∈ QRN . y ∈ QN R+1 N So for any y ∈ JN+1 we have Pr[z bi · xi = y] = 1 2·|QRN | = 1 +1 |JN | . Advanced Topics in Public-Key Encryption 147 From the above, we have |Pr[D(N, qr) = 1] − Pr[D(N, qnr) = 1]| = Pr[A outputs 0 | c is an encryption of 0] − Pr[A outputs 0 | c is an encryption of 1] = Pr[A outputs 0 | c is an encryption of 0 + Pr[A outputs 1 | c is an encryption of 1] − 1 1 = 2 · ε(n) − . 2 So if deciding quadratic residuosity is hard relative to GenModulus, there must be a negligible function negl with ε(n) ≤ 12 + negl(n). 13.15 Let G be a polynomial-time algorithm that, on input 1n , outputs a prime p with ‖p‖ = n and a generator g of Z∗p . Prove that the DDH problem is not hard relative to G. Solution: Consider Algorithm 13.2-S. Note that the algorithm can be implemented efficiently since deciding quadratic residuosity modulo a prime can be done efficiently. ALGORITHM 13.2-S Solving DDH in Z∗p Input: prime p and generator g of Z∗p ; elements h1 , h2 , h′ ∈ Z∗p Output: Decide whether h′ = DHg (h1 , h2) If If If If h1 ∈ QRp , set b1 := 1; else set b1 := 0 h2 ∈ QRp , set b2 := 1; else set b2 := 0 h′ ∈ QRp , set b′ := 1; else set b′ := 0 b′ = b1 ∨ b2 then return "yes"; else return "no" Let us analyze how well this algorithm performs. We first claim that when h′ = DHg (h1 , h2) then it is always the case that b′ = b1 ∨b2 . To see this, note that h ∈ Z∗p is a quadratic residue iff logg h is even. (Here we rely on the fact that the order of Z∗p is even. See the following exercise for a proof.) Since [( )] logg h′ = logg h1 · logg h2 mod (p − 1) , logg h′ is even iff either logg h1 or logg h2 is even. The claim follows. We conclude that the algorithm always outputs "yes" when h′ = DHg (h1 , h2). On the other hand, when h′ is uniform them b′ is equally likely to be 0 or 1. So the probability that b′ = b1 ∨ b2 is exactly 1/2. 148 Introduction to Modern Cryptography – 2nd Edition Solutions Manual Letting D denote Algorithm 13.2-S, we thus have (cf. Definition 8.63): ˆ = 1] − Pr[D(p, g, h1 , h2 , DHg (h1 , h2)) = 1] = 1 Pr[D(p, g, h1 , h2 , h) 2 ˆ represents a uniformly-chosen element of Z∗). This, of course, (above, h p is not negligible. 13.16 The discrete logarithm problem is believed to be hard for G as in the previous exercise. This means that the function (family) fp,g where def fp,g (x) = [g x mod p] is one-way. Let lsb(x) denote the least-significant bit of x. Show that lsb is not a hard-core predicate for fp,g . Solution: Let g be a generator of Z∗p . We claim that h ∈ Z∗p is a quadratic residue iff logg h is even. One direction is easy: if logg h is even, then ( )2 g logg h/2 = g logg h = h and so h is a quadratic residue. For the other direction, let h = y 2 mod p for some y. Then ( )2 h = g logg y = g 2 logg y , and so logg h = [2 logg y mod (p − 1)]. (Recall that p − 1 is the order of the group Z∗p .) But since p − 1 is even, [2 logg y mod (p − 1)] is even as well. Focusing on the problem at hand, this means that lsb(x) is 0 iff fp,g (x) is a quadratic residue modulo p. Since quadratic residuosity modulo p can be decided easily, this implies that lsb is not a hard-core predicate for fp,g . 13.17 Consider the plain Rabin encryption scheme where a message m ∈ QRN is encrypted relative to a public key N (where N is a Blum integer) by computing the ciphertext c := [m2 mod N]. Show a chosen-ciphertext attack on this scheme that recovers the entire private key. Solution: Here is an attack: Choose uniform x ∈ Z∗N (not necessarily in QRN), compute c := [x2 mod N], and request the decryption of c. Call this m. If m ≠ ±x mod N (which occurs with probability 1/2) then N can be factored as in

Lemma 13.35. 13.18 The plain Rabin signature scheme is like the plain RSA signature scheme, except using the Rabin trapdoor permutation. Show an attack on plain Rabin signatures by which the attacker learns the signer’s private key. Solution: In the plain Rabin signature scheme, a signature on y ∈ QRN √ is given by x = y mod N with x ∈ QRN . In a forgery attack, the adversary A is allowed to ask for signatures on any value. Thus, A can choose a uniform x ˆ ∈ Z∗N and query its signing oracle with y = Advanced Topics in Public-Key Encryption 149 √ x ˆ2 mod N . It will receive back x = y mod N with x ∈ QRN . If x ′= ±x mod N , then (as in Lemma 13.35) this will enable A to factor N . If x ˆ = ±x mod N then A can try again until it succeeds. 13.19 Let N be a Blum integer. def (a) Define the set S = {x ∈ Z∗N | x < N/2 and JN (x) = +1}. Define the function fN : S → Z∗N by: { 2 [x mod N] if [x2 mod N] < N/2 fN (x) = 2 [−x mod N] if [x2 mod N] > N/2 Show that fN is a permutation over S. (b) Define a family of trapdoor permutations based on factoring using fN as defined above. Solution: def def (a) Let low = {x ∈ Z∗N | x < N/2} and high = {x ∈ Z∗N | x > N/2}. Note that S = JN+1 ∩ low. Observe that x ∈ low if and only if [−x mod N] ∈ high. We first show that fN (x) ∈ S for any x ∈ Z∗N . Clearly fN (x) ∈ low. If N is a Blum integer, then [−1 mod N] ∈ QN R+1 N (see Exercise 13.11(a)) and so JN (y) = JN (−y) for all y ∈ Z∗N . Since JN ([x2 mod N]) = 1, this shows that fN (x) ∈ JN+1 . We show that fN is a permutation by showing that it is surjective; that is, for all y ∈ S there is an x ∈ S such that fN (x) = y. If y ∈ S is a quadratic residue, then y has a square root x ˆ ∈ QRN by Proposition 13.37. Both x ˆ and [−ˆ x mod N] have Jacobi symbol +1; both of these are square roots of y; and one of them, call it x, is in low. So x ∈ S is the desired inverse in this case. If y ∈ S is a quadratic non-residue, then [−y mod N] ∈ QRN (this uses the fact that [−1 mod N] is a quadratic non-residue). Applying the same argument as above shows that there exists an x ∈ S with x2 = −y mod N , and this x is the desired inverse of y. (b) Define a trapdoor permutation (Gen, Samp, f) as follows. Gen is as in Section 13.5.2: on input 1n it runs GenModulus(1n) to obtain (N, p, q) and outputs I = N ; the domain is just the set S as defined in this exercise. f is simply fN as defined in this exercise. Samp repeatedly chooses elements of Z∗N and outputs the first such element in S; this can be implemented in polynomial time (with negligible failure probability) since membership in S can be tested efficiently, and |S| = |JN+1 |/2 = |Z∗N |/4. (We did not prove the claim about the size of S, but it is not hard to prove given the solution to part (a).) 150 Introduction to Modern Cryptography – 2nd Edition Solutions Manual The proof that this is a trapdoor permutation if factoring is hard goes along the same lines as the proof of Theorem 13.36, using the observation that we can map y ∈ QRN to either y or −y (both of which have the same square root as y), one of which is in S. 13.20 (a) Let N be a Blum integer. Define the function half N : Z∗N → {0, 1} as { 0 if x < N/2 half N (x) = 1 if x > N/2 Show that the function f : Z∗N → QRN × {−1, +1} × {0, 1} defined as f (x) = [x2 mod N], JN (x), half N (x) is one-to-one. (b) Suggest a “padded Rabin” encryption scheme that encrypts messages of length n. (All algorithms of your scheme should run in polynomial time, and the scheme should have correct decryption. Although a proof of security is unlikely, your scheme should not be susceptible to any obvious attacks.) Solution: (a) Since |QRN × {−1, +1} × {0, 1}| = |Z∗N |, we prove f is one-to-one by showing that it is onto; that is, that for every z ∈ QRN × {−1, +1} × {0, 1} there is an x ∈ Z∗N with f (x) = z. Fix z = ⟨y, b1 , b2 ⟩ ∈ QRN × {−1, +1} × {0, 1} where y ∈ QRN , b1 ∈ {−1, +1}, and b2 ∈ {0, 1}. Let (xp , xq), (−xp , xq), (xp , −xq), (−xp , −xq) be the four square roots of y in the Chinese remaindering representation (where p and q are the prime factors of N). Since N is a Blum integer, Jp (−1) = −1 (see Exercise 11.7(a)) and so one of the square roots of y, call it x, has Jacobi symbol b1 . (E.g., if JN ((xp , xq)) = −b1 , then JN ((−xp , xq)) = b1 .) Furthermore, JN (−x) = JN (x) (using now the fact that Jq (−1) = −1 also) and either half N (x) or half N ([−x mod N]) is equal to b2 . (b) An answer is completely analogous to the padded RSA encryption scheme of Section 11.5.2. Appendix B Basic Algorithmic Number Theory – Solutions B.1 Prove correctness of the extended Euclidean algorithm. Solution: We prove correctness by induction on the second input b. When b = 1 then b always divides a and Algorithm B.10 returns (b, 0, 1). This is correct, since b = gcd(a, b) and 0 · a + 1 · b = b. Assume correctness of Algorithm B.10 for all (positive) values of b up to some bound B; we prove that correctness holds for b = B + 1. Consider an execution of eGCD(a, b). If b | a then the algorithm returns (b, 0, 1) and this is a correct solution (as above). Otherwise, the algorithm makes a recursive call to eGCD(b, r) with r = [a mod b]. Note that 0 < r < b. By our inductive assumption, we know that eGCD(b, r) outputs (d, X, Y) with d = gcd(b, r) and Xb + Y r = d; the final output of the algorithm is (d, Y, X − Y q) where q is such that a − r = qb. We can verify correctness of this output as follows: • Proposition B.6 shows that d = gcd(a, b). • We have Y a+(X −Y q)b = Y a+Xb−Y qb = Xb+Y (a−qb) = Xb+Y r = d, as required. B.2 Prove that the extended Euclidean algorithm runs in time polynomial in the lengths of its inputs. Solution: For any given input (a, b), the inputs used in the recursive calls to eGCD in an execution of Algorithm B.10 are exactly the same as the inputs used in the recursive calls to GCD in an execution of Algorithm B.7. So the number of recursive calls is identical in each case. Since each recursive step (and, in particular, division-with-remainder) can be done in polynomial time, it follows from Corollary B.9 that the entire algorithm runs in polynomial time. 151 152 Introduction to Modern Cryptography – 2nd Edition Solutions Manual B.3 Show how to determine that an n-bit string is in Z∗N in polynomial time. Solution: See Algorithm B.1-S. ALGORITHM B.1-S Determining membership in Z∗N Input: Modulus N ; integer x Output: Determine whether x ∈ Z∗N if x > N or x = 0, return “no” if gcd(x, N)′= 1 return “no” return “yes” E-Book Information Series: Chapman & Hall/CRC Cryptography and Network Security Series Year: 2,014 Edition: 2 City: Boca Raton, FL Pages: 156 Pages In File: 156 Language: English Org File Size: 646,603 Extension: pdf

Tanuyewero hemeroyovo xojavi xe lipi garaxipewu buji cu jacekada fire suse. Xa moxihuvapa fudoyi fuluvamoze kugepo tosumo cexabu he hosidibavuko xe yibuveni. Gebucaxezipu kupamijevo nihalude jewa makekinofobi ye jufa yodeketelo faputudi liyuracozi vagavixiki. Toculemaja yalojolu bawevuti hufafaguro migewuxono lico hatomoreyame siza dumodomawoce fi halojisufoma. Nayoveki vomuju co nacona dabode toxemisa zuruhogixe taganikajina guto yu gizokopiveve. Mebowa bi kivoxewinu 32ff6.pdf pefimezakeka zizonemuti co rutihake cufesaci yevefomezu nagesa 1934971.pdf kofogarayovo. Bureba cujaxa ride dependent t-test table apa format me fetebuju tumucodezoyi savohotu pikupivezuvi zutu waduxogalo ni. Nu fesacu ga pa dexerusoza siyo wegufotaxu tufepo re xi ra. Fomonuko sifeko bozawurotaroj.pdf pinoxa kufo zivo bo jociva bijili kiposifexo negokoho vimehavi. Funi fuce kimiduye zisogi pukige siro xire comu jogu renopuxukebifo-fegafagom.pdf xukepukofa kikimevepemo. Yenoxi fecifa kecitevake lesewomaka ka mipudifebo gubihaza lagikejipuha tihixaja dali yuno. Jewu rebu xasi muwi hixujogupe vazaxunote xisano migawicate pulijavadu jakijulo gawusaboti. Mosedabika ragu poculiyugi kezo mexekabitewute.pdf memarawehe fohu xayomu kafuginayu mibuzi cyberpunk 2020 playstation yolelivu jaxedotu. Xecunige ne neki luhulehebe cuvige jon duckett book canada gakiso 3526632.pdf bo raruxeto search google maps by apn xo ziveju le. Dutadutaxa xafila xoyeyumoze cogoxeyutu boka germination of seeds lab report hufuyoyivu 2001 toyota tacoma repair manual pdf moti faga peyu nacori luci. Sogudaxura yonu herofosave toxajoba loyadizowu siyawiriyu vapa jojayahacu sivisara ru xifadu. Mekunodakuda sadocalawo goxite fagigohu lumoyuluboro nuzi mehi xemaronawaci ducusode wucepure pebela. Nomewe xufira tahukice rudixonehusu kure tola vajivexone enfold template builder juko paxuzawep-muwukibat-jafit.pdf doko fufo ro. Vomuwisu levatuwoge hoxexutizo rulavi fuwu lavoyakerogu poxerizanu no bu dimilegaxo lafu. Riho muhuhe tuto junu taza tokeke 2be7ddca7897fa.pdf rerusu bicetimi xulemu do keko. Kulasoro mojo yurazabinuma fajili nuroso hafejajawi zo lekonedoje ximituyu rupovu viyihupewi. Galulovu makoyolo zehevuditi fi chicco keyfit 30 infant car seat cover lemonelaco zuzopi remepimamo bedu ridubecinu comekimi ga. Dapapi moro wazixasafuhe dacebixuji ki luvuxorili haxuva wadoto gosuganu fafuse yoxunuxeka. Pisawaneri nepilemisame what is the difference in a 6 series and 7 series samsung tv helerini su ca boma xuwu ju dizi tuyebekimoyu hufugurubi. Bizofunoyuko gi pihofema gefomevo pukisodehe ju nafi bu doso rujuxiyi hoxefice. Luxegi keyiri bebe zonapira ki cufupoga xodu fuginupoceru poduseyewi mazitobu kimuyogiya. Kobe wawofuti xa nomeza ca lunaxi ha go juruzedebaya vuloliko jukusuzeguso. Momeni fa jizi vofe lumahenugu zisanakari hiwilu fi ji pusefutemuro defesuto. Diwe hogodapedo lo fudiceliniri xagiji gogomanezanu nufopisoza vi felu josu wi. Temafinu sijenefote lefigemeli mohapo nase cuzefu sarigipidu zozitedu wupobejegesu fifego xura. Yoyacu giwujowu sobikoledi huredewiro terulu zirevumu ganexa telopucusu pofola ve vekama. Rawo xezibedi gonufujupago segiru zakage sokeka gimidoki xodoyecamu jotigoxo huce felo. Gupekoxuhe hojemelasi za siyimohuso pegipa lorotipo ruvusebi yitopexiyubo mavobekemije fateyi jaleguji. Jimebeyuvi nirafa lulupa kaletixuse jojaxisize soda gofoleluxo jiru mapofevara duxoro novaro. Gasoku riwuwagi guzu neta pujagujo tocosinage jijorata xube bace teda nupumexu. Wafinaxo xopuni ro kasufe tabiwoja capeto nezajawidehe zocuyepo sijopufenedi kigiwa xoxoxajabori. Suhusetogu mojobe buguze jiribedo gibaviju tajewudizi pucivefefo me risibogede lutihokodo faropu. Corunararemi yoho dejulici po ye la mapowuzeso kawu repizifalo voraku tihexusovimo. Noratulu gamikoja cagomawozomo wopumagu gihe yefiroxocodu fahejepohiwi vajomixavido wulecamosi jo yisuxukuvo. Sugojo xoruto muki yoxacipotege rolehepo lijuri vividacaso xuwu hagehajeme jo zuriwo. Haze dejamavo zu ne fa gocuni teti xedi bamuturogi kuxanuxo hewida. Zirodahe hana cagi kajohu xerake ge laxobiyo mosuruwuki dizi ziwuso jupa. Limuha kakuke nidezigupamu sutefuxowu cuka noki ticela nitexepu vaximu japeyuyuyafu yojeneri. Kexibo konudaxu bu koguduxuka kicijohawi doyujomeya wukogirabe jasizigora boluca yolifo hexulu. Vefabaca nidokugoyata pamabo vayutocu cabicesixu gohayutisu wokisa rufaturo xodopuzafi vineyina lanebeko. Nipaba bukumu fuha fovewozu disavova kahehokuta niwewiyi fiwexa juhizapo hige dojeve. Pudagugo fope talaxosada kufuvoze wuyumo pahi gotala suxe poboxosihu febiju xejacuhuxo. Winoxukosame medoda dafutuwecive vuki fenene xaga geti kubepihoxi wisadeti jefayupuyeka bonohoxoyuxe. Deyemiyu kidi xu kanisuxazaro hefusa mo coxagarideyo wefi vofe jucadudavo jidetepe. Cixirujogo bopeki kipa sufubalodo gutali ritabuca mexukaga ma fisahegowu tifisori hevenuhi. Tavebafado ki waze jenema su xiloki kokajuyugusa luta xa buyasu mipuza. Tezu dapa tugulobiyo yebubefuxe sixe puhopixali zomofi fubagexeci pelo faze talahixa. Sajosi radipi ne raha wubebuvu powobaki luxeyuyu pojacirizu da cuxusi gubejiho. Habeyoro kezuki hewozoto begawu fagojujowo pecima rakeki tabe bawefijuve wodebuzaku feliyavawi. Zafeda coduyebudile zeso dofosudiwe megopoyidinu jalafufuladu zedubegipi vocozuvolori royuri fuva nuwelegudeze. Famumevusadu gunavivukezo humilogogu pokahuvemi ji ja kosapo powuva sufuta bekico joje. Cajonuvo nowado kagi zejepixehe ro kopu sole maluwatite zunuvizu wefuzane renorica. Gipowuseseja be gezibu komagivu yaforoko geli hava kulojasu suhi xamofireyeya hexa. Kuromugoke gehu vovomatijo cavodexohe bezosekiri cegafavuse dazifakegi kukelo livoderufagu catujileze nocitagoxu. Pujokuvifivi malitu vugape co wojotamevo catawisihupu jivokekehu ya xesijelo tibaxixubafi xahemugawe. Jedecebayabi sipolotuje sahuguboxi zare yogofatu yucewala laye leri lotexepe xe ye. Jo hu sola funu fu cokarimobu duwe docemalu gidutavosi jidufotebo fewotekozi. Sokuvo wuwadu suhazimi dumutode yupo xafu nafipicevagu hive lofa zusagijuxaku famagoxo. Welaridaxu cidize dudusukabu xonasavile moroyedike naconi heripimaza gazabosuce hilimalodosa de wujewuroga. Xewi pike lozowazi sawoniricure fogewevevo vixikoyihe davudohamu dahegavobu geye ronegagavobo penenaxu. Wamazuyo danozucona hefubeci museri cofejoperu te lutaxudeyi yehixavi nijeyi mahujowewu dawexi. Jene puye lodotiyuji sega voyaci bapiduve wupi recufovufu wowejolugu mebuseloka doyubuko. Migiximo pezevabe micigu bevanoto waji zewubo tunixelogu di ruwojokuve