Understanding Cryptography

A Textbook for Students and Practitioners

Foreword by Bart Preneel

Prof Dr Ing Christof Paar Chair for Embedded Security

Department of Electrical Engineering and Information Sciences

Ruhr Universita t Bochum 0 Bochum

escrypt GmbH Embedded Security Zentrum fu r IT Sicherheit

ISBN 9 6 6 e ISBN 9 6 3 DOI 0 6 3

Springer Heidelberg Dordrecht London New York

ACM Computing Classification E 3 K K

Library of Congress Control Number 47 c Springer Verlag Berlin Heidelberg

This work is subject to copyright All rights are reserved whether the whole or part of the material is concerned specifically the rights of translation reprinting reuse of illustrations recitation broadcasting reproduction on microfilm or in any other way and storage in data banks Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9 in its current version and permission for use must always be obtained from Springer Violations are liable to prosecution under the German Copyright Law

The use of general descriptive names registered names trademarks etc in this publication does not imply even in the absence of a specific statement that such names are exempt from the relevant protective laws and regulations and therefore free for general use

Cover design KuenkelLopka GmbH Printed on acid free paper

Springer is part of Springer Science Business Media www springer com

Flora Maja Noah and Sarah as well as to

While writing this book we noticed that for some reason the names of our spouses and children are limited to five letters As far as we know this has no cryptographic relevance

Academic research in cryptology started in the mid s today it is a mature re search discipline with an established professional organization IACR International Association for Cryptologic Research thousands of researchers and dozens of in ternational conferences Every year more than a thousand scientific papers are pub lished on cryptology and its applications

Until the s cryptography was almost exclusively found in diplomatic mili tary and government applications During the s the financial and telecommuni cations industries deployed hardware cryptographic devices The first mass market cryptographic application was the digital mobile phone system of the late s Today everyone uses cryptography on a daily basis Examples include unlocking a car or garage door with a remote control device connecting to a wireless LAN buying goods with a credit or debit card in a brick and mortar store or on the Inter net installing a software update making a phone call via voice over IP or paying for a ride on a public transport system There is no doubt that emerging application areas such as e health car telematics and smart buildings will make cryptography even more ubiquitous

Cryptology is a fascinating discipline at the intersection of computer science mathematics and electrical engineering As cryptology is moving fast it is hard to keep up with all the developments During the last 25 years the theoretical foun dations of the area have been strengthened we now have a solid understanding of security definitions and of ways to prove constructions secure Also in the area of applied cryptography we witness very fast developments old algorithms are broken and withdrawn and new algorithms and protocols emerge

While several excellent textbooks on cryptology have been published in the last decade they tend to focus on readers with a strong mathematical background More over the exciting new developments and advanced protocols form a temptation to add ever more fancy material It is the great merit of this textbook that it restricts itself to those topics that are relevant to practitioners today Moreover the mathe matical background and formalism is limited to what is strictly necessary and it is introduced exactly in the place where it is needed This less is more approach is very suitable to address the needs of newcomers in the field as they get introduced

step by step to the basic concepts and judiciously chosen algorithms and protocols Each chapter contains very helpful pointers to further reading for those who want to expand and deepen their knowledge

Overall I am very pleased that the authors have succeeded in creating a highly valuable introduction to the subject of applied cryptography I hope that it can serve as a guide for practitioners to build more secure systems based on cryptography and as a stepping stone for future researchers to explore the exciting world of cryptog raphy and its applications

Leuven August Bart Preneel

Cryptography has crept into everything from Web browsers and e mail programs to cell phones bank cards cars and even into medical implants In the near fu ture we will see many new exciting applications for cryptography such as radio frequency identification RFID tags for anti counterfeiting or car to car commu nications we ve worked on securing both of these applications This is quite a change from the past where cryptography had been traditionally confined to very specific applications especially government communications and banking systems As a consequence of the pervasiveness of crypto algorithms an increasing number of people must understand how they work and how they can be applied in prac tice This book addresses this issue by providing a comprehensive introduction to modern applied cryptography that is equally suited for students and practitioners in industry

Our book provides the reader with a deep understanding of how modern cryp tographic schemes work We introduce the necessary mathematical concepts in a way that is accessible for every reader with a minimum background in college level calculus It is thus equally well suited as a textbook for undergraduate or begin ning graduate classes or as a reference book for practicing engineers and computer scientists who are interested in a solid understanding of modern cryptography

The book has many features that make it a unique source for practitioners and stu dents We focused on practical relevance by introducing most crypto algorithms that are used in modern real world applications For every crypto scheme up to date se curity estimations and key length recommendations are given We also discuss the important issue of software and hardware implementation for every algorithm In addition to crypto algorithms we introduce topics such as important cryptographic protocols modes of operation security services and key establishment techniques Many very timely topics e g lightweight ciphers which are optimized for con strained applications such as RFID tags or smart cards or new modes of operations are also contained in the book

A discussion section at the end of each chapter with annotated references pro vides plenty of material for further reading For classroom use these sections are

an excellent source for course projects In particular when used as a textbook the companion website for the book is highly recommended

Readers will find many ideas for course projects links to open source software test vectors and much more information on contemporary cryptography In addition links to video lectures are provided

The material in this book has evolved over many years and is classroom proven We ve taught it both as a course for beginning graduate students and advanced un dergraduate students and as a pure undergraduate course for students majoring in our IT security programs We found that one can teach most of the book content in a two semester course with 90 minutes of lecture time plus 45 minutes of help session with exercises per week total of 10 ECTS credits In a typical US style three credit course or in a one semester European course some of the material should be omitted Here are some reasonable choices for a one semester course

Curriculum 1 Focus on the application of cryptography e g in a computer sci ence or electrical engineering program This crypto course is a good addition to courses in computer networks or more advanced security courses Sect

Curriculum 2 Focus on cryptographic algorithms and their mathematical back ground e g as an applied cryptography course in computer science electrical engi neering or in an undergraduate math program This crypto course works also nicely as preparation for a more theoretical graduate courses in cryptography 0

Trained as engineers we have worked in applied cryptography and security for more than 15 years and hope that the readers will have as much fun with this fasci nating field as we ve had

Bochum Christof Paar

September Jan Pelzl

Writing this book would have been impossible without the help of many people We hope we did not forget anyone in our list

We are grateful for the excellent work of Daehyun Strobel and Pascal Wi mann who provided most of the artwork in the book and never complained about our many changes Axel Poschmann provided the section about the PRESENT block cipher a very timely topic and we are thankful for his excellent work Help with technical questions was provided by Frederick Armknecht stream ciphers Roberto Avanzi finite fields and elliptic curves Alexander May number theory Alfred Menezes and Neal Koblitz history of elliptic curve cryptography Matt Robshaw AES and Damian Weber discrete logarithms

Many thanks go the members of the Embedded Security group at the Univer sity of Bochum Andrey Bogdanov Benedikt Driessen Thomas Eisenbarth Tim Gu neysu Stefan Heyse Markus Kasper Timo Kasper Amir Moradi and Daehyun Strobel who did much of the technical proofreading and provided numerous sug gestions for improving the presentation of the material Special thanks to Daehyun

for helping with examples and some advanced LATEX work and to Markus for his help with problems Olga Paustjan s help with artwork and typesetting is also very much appreciated

An earlier generation of doctoral students from our group Sandeep Kumar Kerstin Lemke Rust Andy Rupp Kai Schramm and Marko Wolf helped to cre ate an online course that covered similar material Their work was very useful and was a great inspiration when writing the book

Bart Preneel s willingness to provide the Foreword is a great honor for us and we would like to thank him at this point again Last but not least we thank the people from Springer for their support and encouragement In particular thanks to our editor Ronan Nugent and to Alfred Hofmann

Introduction to Cryptography and Data Security

This section will introduce the most important terms of modern cryptology and will teach an important lesson about proprietary vs openly known algorithms We will also introduce modular arithmetic which is also of major importance in public key cryptography

In this chapter you will learn

The general rules of cryptography

Key lengths for short medium and long term security

The difference between different types of attacks against ciphers

A few historical ciphers and on the way we will learn about modular arithmetic which is of major importance for modern cryptography as well

Why one should only use well established encryption algorithms

C Paar J Pelzl Understanding Cryptography 1

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

Overview of Cryptology and This Book

If we hear the word cryptography our first associations might be e mail encryption secure website access smart cards for banking applications or code breaking during World War II such as the famous attack against the German Enigma encryption machine Fig 1

The German Enigma encryption machine reproduced with permission from the Deutsches Museum Munich

Cryptography seems closely linked to modern electronic communication How ever cryptography is a rather old business with early examples dating back to about B C when non standard secret hieroglyphics were used in ancient Egypt Since Egyptian days cryptography has been used in one form or the other in many if not most cultures that developed written language For instance there are doc umented cases of secret writing in ancient Greece namely the scytale of Sparta Fig 2 or the famous Caesar cipher in ancient Rome about which we will learn later in this chapter This book however strongly focuses on modern cryptographic

methods and also teaches many data security issues and their relationship with cryp tography

Let s now have a look at the field of cryptography Fig 3 The first thing

Overview of Cryptology and This Book 3

Overview of the field of cryptology

that we notice is that the most general term is cryptology and not cryptography Cryptology splits into two main branches

Cryptography is the science of secret writing with the goal of hiding the mean ing of a message

Cryptanalysis is the science and sometimes art of breaking cryptosystems You might think that code breaking is for the intelligence community or perhaps or ganized crime and should not be included in a serious classification of a scien tific discipline However most cryptanalysis is done by respectable researchers in academia nowadays Cryptanalysis is of central importance for modern cryp tosystems without people who try to break our crypto methods we will never know whether they are really secure or not See Sect for more discussion about this issue

Because cryptanalysis is the only way to assure that a cryptosystem is secure it is an integral part of cryptology Nevertheless the focus of this book is on cryptography We introduce most important practical crypto algorithms in detail These are all crypto algorithms that have withstood cryptanalysis for a long time in most cases for several decades In the case of cryptanalysis we will mainly restrict ourselves to providing state of the art results with respect to breaking the crypto al gorithms that are introduced e g the factoring record for breaking the RSA scheme Let s now go back to Fig Cryptography itself splits into three main branches

Symmetric Algorithms are what many people assume cryptography is about two parties have an encryption and decryption method for which they share a secret key All cryptography from ancient times until was exclusively based on symmetric methods Symmetric ciphers are still in widespread use especially for data encryption and integrity check of messages

Asymmetric or Public Key Algorithms In an entirely different type of cipher was introduced by Whitfield Diffie Martin Hellman and Ralph Merkle In public key cryptography a user possesses a secret key as in symmetric cryptog raphy but also a public key Asymmetric algorithms can be used for applications such as digital signatures and key establishment and also for classical data en cryption

Cryptographic Protocols Roughly speaking crypto protocols deal with the ap plication of cryptographic algorithms Symmetric and asymmetric algorithms

can be viewed as building blocks with which applications such as secure Inter net communication can be realized The Transport Layer Security TLS scheme which is used in every Web browser is an example of a cryptographic protocol

Strictly speaking hash functions which will be introduced in 1 form a third class of algorithms but at the same time they share some properties with symmetric ciphers

In the majority of cryptographic applications in practical systems symmetric and asymmetric algorithms and often also hash functions are all used together This is sometimes referred to as hybrid schemes The reason for using both families of algorithms is that each has specific strengths and weaknesses

The main focus of this book is on symmetric and asymmetric algorithms as well as hash functions However we will also introduce basic security protocols In particular we will introduce several key establishment protocols and what can be achieved with crypto protocols confidentiality of data integrity of data authentica tion of data user identification etc

Symmetric Cryptography

This section deals with the concepts of symmetric ciphers and it introduces the historic substitution cipher Using the substitution cipher as an example we will learn the difference between brute force and analytical attacks

Symmetric cryptographic schemes are also referred to as symmetric key secret key and single key schemes or algorithms Symmetric cryptography is best introduced with an easy to understand problem There are two users Alice and Bob who want to communicate over an insecure channel Fig 4 The term channel might sound a bit abstract but it is just a general term for the communication link This can be the Internet a stretch of air in the case of mobile phones or wireless LAN communica tion or any other communication media you can think of The actual problem starts with the bad guy Oscar1 who has access to the channel for instance by hacking into an Internet router or by listening to the radio signals of a Wi Fi communica tion This type of unauthorized listening is called eavesdropping Obviously there are many situations in which Alice and Bob would prefer to communicate without Oscar listening For instance if Alice and Bob represent two offices of a car man ufacturer and they are transmitting documents containing the business strategy for the introduction of new car models in the next few years these documents should

1 The name Oscar was chosen to remind us of the word opponent

not get into the hands of their competitors or of foreign intelligence agencies for that matter

Communication over an insecure channel

In this situation symmetric cryptography offers a powerful solution Alice en crypts her message x using a symmetric algorithm yielding the ciphertext y Bob receives the ciphertext and decrypts the message Decryption is thus the inverse process of encryption Fig What is the advantage If we have a strong encryp tion algorithm the ciphertext will look like random bits to Oscar and will contain no information whatsoever that is useful to him

Symmetric key cryptosystem

The variables x y and k in Fig are important in cryptography and have special names

x is called plaintext or cleartext

y is called ciphertext

the set of all possible keys is called the key space

The system needs a secure channel for distribution of the key between Alice and Bob The secure channel shown in Fig can for instance be a human who is transporting the key in a wallet between Alice and Bob This is of course a somewhat cumbersome method An example where this method works nicely is the pre shared keys used in Wi Fi Protected Access WPA encryption in wireless

LANs Later in this book we will learn methods for establishing keys over insecure channels In any case the key has only to be transmitted once between Alice and Bob and can then be used for securing many subsequent communications

One important and also counterintuitive fact in this situation is that both the en cryption and the decryption algorithms are publicly known It seems that keeping the encryption algorithm secret should make the whole system harder to break However secret algorithms also mean untested algorithms The only way to find out whether an encryption method is strong i e cannot be broken by a determined attacker is to make it public and have it analyzed by other cryptographers Please see Sect for more discussion on this topic The only thing that should be kept secret in a sound cryptosystem is the key

Of course if Oscar gets hold of the key he can easily decrypt the message since the algorithm is publicly known Hence it is crucial to note that the problem of transmitting a message securely is reduced to the problems of transmitting a key secretly and of storing the key in a secure fashion

In this scenario we only consider the problem of confidentiality that is of hiding the contents of the message from an eavesdropper We will see later in this book that there are many other things we can do with cryptography such as preventing Oscar from making unnoticed changes to the message message integrity or assuring that a message really comes from Alice sender authentication

Simple Symmetric Encryption The Substitution Cipher

We will now learn one of the simplest methods for encrypting text the substitution replacement cipher Historically this type of cipher has been used many times and it is a good illustration of basic cryptography We will use the substitution cipher for learning some important facts about key lengths and about different ways of attacking ciphers

The goal of the substitution cipher is the encryption of text as opposed to bits in modern digital systems The idea is very simple We substitute each letter of the alphabet with another one

For instance the pop group ABBA would be encrypted as kddk

We assume that we choose the substitution table completely randomly so that an attacker is not able to guess it Note that the substitution table is the key of this cryptosystem As always in symmetric cryptography the key has to be distributed between Alice and Bob in a secure fashion

Example Let s look at another ciphertext

iq ifcc vqqr fb rdq vfllcq na rdq cfjwhwz hr bnnb hcc hwwhbsqvqbre hwq vhlq

This does not seem to make too much sense and looks like decent cryptography However the substitution cipher is not secure at all Let s look at ways of breaking the cipher

First Attack Brute Force or Exhaustive Key Search

Brute force attacks are based on a simple concept Oscar the attacker has the ci phertext from eavesdropping on the channel and happens to have a short piece of plaintext e g the header of a file that was encrypted Oscar now simply decrypts the first piece of ciphertext with all possible keys Again the key for this cipher is the substitution table If the resulting plaintext matches the short piece of plaintext he knows that he has found the correct key

In practice a brute force attack can be more complicated because incorrect keys can give false positive results We will address this issue in Sect

It is important to note that a brute force attack against symmetric ciphers is al ways possible in principle Whether it is feasible in practice depends on the key space i e on the number of possible keys that exist for a given cipher If testing all the keys on many modern computers takes too much time i e several decades the cipher is computationally secure against a brute force attack

Let s determine the key space of the substitution cipher When choosing the re placement for the first letter A we randomly choose one letter from the 26 letters of the alphabet in the example above we chose k The replacement for the next al phabet letter B was randomly chosen from the remaining 25 letters etc Thus there exist the following number of different substitution tables

key space of the substitution cipher 26 25 3 2 1 26

Even with hundreds of thousands of high end PCs such a search would take several decades Thus we are tempted to conclude that the substitution cipher is secure But this is incorrect because there is another more powerful attack

Second Attack Letter Frequency Analysis

First we note that the brute force attack from above treats the cipher as a black box i e we do not analyze the internal structure of the cipher The substitution cipher can easily be broken by such an analytical attack

The major weakness of the cipher is that each plaintext symbol always maps to the same ciphertext symbol That means that the statistical properties of the plaintext are preserved in the ciphertext If we go back to the second example we observe that the letter q occurs most frequently in the text From this we know that q must be the substitution for one of the frequent letters in the English language

For practical attacks the following properties of language can be exploited

Determine the frequency of every ciphertext letter The frequency distribution often even of relatively short pieces of encrypted text will be close to that of the given language in general In particular the most frequent letters can often easily be spotted in ciphertexts For instance in English E is the most frequent letter about 13 T is the second most frequent letter about 9 A is the third most frequent letter about 8 and so on Table lists the letter frequency distribution of English

The method above can be generalized by looking at pairs or triples or quadru ples and so on of ciphertext symbols For instance in English and some other European languages the letter Q is almost always followed by a U This behavior can be exploited to detect the substitution of the letter Q and the letter U

If we assume that word separators blanks have been found which is only some

times the case one can often detect frequent short words such as THE AND etc Once we have identified one of these words we immediately know three letters or whatever the length of the word is for the entire text

In practice the three techniques listed above are often combined to break substi tution ciphers

Example If we analyze the encrypted text from Example we obtain

WE WILL MEET IN THE MIDDLE OF THE LIBRARY AT NOON ALL ARRANGEMENTS ARE MADE

Relative letter frequencies of the English language

Lesson learned Good ciphers should hide the statistical properties of the encrypted plaintext The ciphertext symbols should appear to be random Also a large key space alone is not sufficient for a strong encryption function

This section deals with recommended key lengths of symmetric ciphers and differ ent ways of attacking crypto algorithms It is stressed that a cipher should be secure even if the attacker knows the details of the algorithm

General Thoughts on Breaking Cryptosystems

If we ask someone with some technical background what breaking ciphers is about he she will most likely say that code breaking has to do with heavy mathematics smart people and large computers We have images in mind of the British code breakers during World War II attacking the German Enigma cipher with extremely smart mathematicians the famous computer scientist Alan Turing headed the ef forts and room sized electro mechanical computers However in practice there are also other methods of code breaking Let s look at different ways of breaking cryp tosystems in the real world Fig 6

Overview of cryptanalysis

Classical Cryptanalysis

Classical cryptanalysis is understood as the science of recovering the plaintext x from the ciphertext y or alternatively recovering the key k from the ciphertext y We recall from the earlier discussion that cryptanalysis can be divided into ana lytical attacks which exploit the internal structure of the encryption method and brute force attacks which treat the encryption algorithm as a black box and test all possible keys

Implementation Attacks

Side channel analysis can be used to obtain a secret key for instance by measuring the electrical power consumption of a processor which operates on the secret key The power trace can then be used to recover the key by applying signal processing techniques In addition to power consumption electromagnetic radiation or the run time behavior of algorithms can give information about the secret key and are thus useful side channels 2 Note also that implementation attacks are mostly relevant against cryptosystems to which an attacker has physical access such as smart cards In most Internet based attacks against remote systems implementation attacks are usually not a concern

Social Engineering Attacks

Bribing blackmailing tricking or classical espionage can be used to obtain a secret key by involving humans For instance forcing someone to reveal his her secret key e g by holding a gun to his her head can be quite successful Another less violent attack is to call people whom we want to attack on the phone and say This is

2 Before you switch on the digital oscilloscope in your lab in order to reload your Geldkarte the Geldkarte is the electronic wallet function integrated in most German bank cards to the maximum amount of e Modern smart cards have built in countermeasures against side channel attacks and are very hard to break

the IT department of your company For important software updates we need your password It is always surprising how many people are na ve enough to actually give out their passwords in such situations

This list of attacks against cryptographic system is certainly not exhaustive For instance buffer overflow attacks or malware can also reveal secret keys in software systems You might think that many of these attacks especially social engineering and implementation attacks are unfair but there is little fairness in real world cryptography If people want to break your IT system they are already breaking the rules and are thus unfair The major point to learn here is

An attacker always looks for the weakest link in your cryptosystem That means we have to choose strong algorithms and we have to make sure that social engineering and implementation attacks are not practical

Even though both implementation attacks and social engineering attacks can be quite powerful in practice this book mainly assumes attacks based on mathematical cryptanalysis

Solid cryptosystems should adhere to Kerckhoffs Principle postulated by Au guste Kerckhoffs in

Important Remark Kerckhoffs Principle is counterintuitive It is extremely tempt ing to design a system which appears to be more secure because we keep the details hidden This is called security by obscurity However experience and military his tory has shown time and again that such systems are almost always weak and they are very often broken easily as soon as the secret design has been reverse engineered or leaked out through other means An example is the Content Scrambling System CSS for DVD content protection which was broken easily once it was reverse engineered This is why a cryptographic scheme must remain secure even if its de scription becomes available to an attacker

How Many Key Bits Are Enough

During the s there was much public discussion about the key length of ciphers Before we provide some guidelines there are two crucial aspects to remember

The discussion of key lengths for symmetric crypto algorithms is only relevant if a brute force attack is the best known attack As we saw in Sect during the security analysis of the substitution cipher if there is an analytical attack that

works a large key space does not help at all Of course if there is the possibility of social engineering or implementation attacks a long key also does not help

The key lengths for symmetric and asymmetric algorithms are dramatically dif ferent For instance an 80 bit symmetric key provides roughly the same security as a bit RSA RSA is a popular asymmetric algorithm key

Both facts are often misunderstood especially in the semitechnical literature

Table gives a rough indication of the security of symmetric ciphers with re spect to brute force attacks As described in Sect a large key space is a nec essary but not sufficient condition for a secure symmetric cipher The cipher must also be strong against analytical attacks

Estimated time for successful brute force attacks on symmetric algorithms with different key lengths

Foretelling the Future Of course predicting the future tends to be tricky We can t really foresee new technical or theoretical developments with certainty As you can imagine it is very hard to know what kinds of computers will be available in the year For medium term predictions Moore s Law is often assumed Roughly speaking Moore s Law states that computing power doubles every 18 months while the costs stay constant This has the following implications in cryptography If today we need one month and computers worth 00 to break a cipher X then

The cost for breaking the cipher will be 0 in 18 months since we only have to buy half as many computers

0 in years and so on

It is important to stress that Moore s Law is an exponential function In 15 years i e after 10 iterations of computer power doubling we can do as many computations for the same money we would need to spend today Stated differently we only need to spend about 1 th of today s money to do the same computation In the example above that means that we can break cipher X in 15 years within one month at a cost of about 1 24 Alternatively with 00 an attack can be accomplished within 45 minutes in 15 years from now Moore s Law behaves similarly to a bank account with a 50 interest rate The compound interest grows very very quickly Unfortunately there are few trustworthy banks which offer such an interest rate

Modular Arithmetic and More Historical Ciphers

In this section we use two historical ciphers to introduce modular arithmetic with integers Even though the historical ciphers are no longer relevant modular arith metic is extremely important in modern cryptography especially for asymmetric algorithms Ancient ciphers date back to Egypt where substitution ciphers were used A very popular special case of the substitution cipher is the Caesar cipher which is said to have been used by Julius Caesar to communicate with his army The Caesar cipher simply shifts the letters in the alphabet by a constant number of steps When the end of the alphabet is reached the letters repeat in a cyclic way similar to numbers in modular arithmetic

To make computations with letters more practicable we can assign each letter of the alphabet a number By doing so an encryption with the Caesar cipher simply becomes a modular addition with a fixed value Instead of just adding constants a multiplication with a constant can be applied as well This leads us to the affine cipher

Both the Caesar cipher and the affine cipher will now be discussed in more detail

Almost all crypto algorithms both symmetric ciphers and asymmetric ciphers are based on arithmetic within a finite number of elements Most number sets we are used to such as the set of natural numbers or the set of real numbers are infinite In the following we introduce modular arithmetic which is a simple way of performing arithmetic in a finite set of integers

Let s look at an example of a finite set of integers from everyday life

Example Consider the hours on a clock If you keep adding one hour you ob tain

1h 2h 3h 11h 12h 1h 2h 3h 11h 12h 1h 2h 3h

Even though we keep adding one hour we never leave the set

Let s look at a general way of dealing with arithmetic in such finite sets

Example We consider the set of the nine numbers

0 1 2 3 4 5 6 7 8

We can do regular arithmetic as long as the results are smaller than 9 For instance

But what about 8 4 Now we try the following rule Perform regular integer arith metic and divide the result by 9 We then consider only the remainder rather than the original result Since 8 4 12 and has a remainder of 3 we write

We now introduce an exact definition of the modulo operation

There are a few implications from this definition which go beyond the casual rule divide by the modulus and consider the remainder We discuss these implications below

Computation of the Remainder

It is always possible to write a Z such that

a q m r for 0 r m

Since a r q m m divides a r we can now write a r mod m Note that

r 0 1 2 m 1

Example Let a 42 and m 9 Then

and therefore 42 6 mod 9

The Remainder Is Not Unique

It is somewhat surprising that for every given modulus m and number a there are infinitely many valid remainders Let s look at another example

Example We want to reduce 12 modulo 9 Here are several results which are correct according to the definition

12 3 mod 9 3 is a valid remainder since 9 12 3

12 21 mod 9 21 is a valid remainder since 9 21 3

12 6 mod 9 6 is a valid remainder since 9 6 3

where the x y means x divides y There is a system behind this behavior The set of numbers

24 15 6 3 12 15 24

form what is called an equivalence class There are eight other equivalence classes for the modulus 9

27 18 9 0 9 18 27

26 17 8 1 10 19 28

19 10 1 8 17 26 35

All Members of a Given Equivalence Class Behave Equivalently

For a given modulus m it does not matter which element from a class we choose for a given computation This property of equivalent classes has major practical implications If we have involved computations with a fixed modulus which is usually the case in cryptography we are free to choose the class element that results in the easiest computation Let s look first at an example

Example The core operation in many practical public key schemes is an expo nentiation of the form xe mod m where x e m are very large integers say bits each Using a toy size example we can demonstrate two ways of doing modular ex ponentiation We want to compute 38 mod 7 The first method is the straightforward approach and for the second one we switch between equivalent classes

38 2 mod 7 since 9 2

Note that we obtain the fairly large intermediate result even though we know that our final result cannot be larger than 6

Here is a much smarter method First we perform two partial exponentiations

38 34 34 81 81

We can now replace the intermediate results 81 by another member of the same equivalence class The smallest positive member modulo 7 in the class is 4 since 81 11 7 4 Hence

38 81 81 4 4 16 mod 7

From here we obtain the final result easily as 16 2 mod 7

Note that we could perform the second method without a pocket calculator since the numbers never become larger than 81 For the first method on the other hand dividing by 7 is mentally already a bit challenging As a general rule we should remember that it is almost always of computational advantage to apply the modulo reduction as soon as we can in order to keep the numbers small

Of course the final result of any modulo computation is always the same no matter how often we switch back and forth between equivalent classes

Which Remainder Do We Choose

By agreement we usually choose r in Eq such that

However mathematically it does not matter which member of an equivalent class we use

After studying the properties of modulo reduction we are now ready to define in more general terms a structure that is based on modulo arithmetic Let s look at the mathematical construction that we obtain if we consider the set of integers from

zero to m 1 together with the operations addition and multiplication

Let s first look at an example for a small integer ring

Example Let m 9 i e we are dealing with the ring Z9 0 1 2 3 4 5 6 7 8 Let s look at a few simple arithmetic operations

More about rings and finite fields which are related to rings is discussed in Sect At this point the following properties of rings are important

We can add and multiply any two numbers and the result is always in the ring A ring is said to be closed

Addition and multiplication are associative e g a b c a b c and

a b c a b c for all a b c Zm

There is the neutral element 0 with respect to addition i e for every element

a Zm it holds that a 0 a mod m

For any element a in the ring there is always the negative element a such that

a a 0 mod m i e the additive inverse always exists

There is the neutral element 1 with respect to multiplication i e for every ele ment a Zm it holds that a 1 a mod m

The multiplicative inverse exists only for some but not for all elements Let

a Z the inverse a 1 is defined such that

If an inverse exists for a we can divide by this element since b a b a 1 mod

It takes some effort to find the inverse usually employing the Euclidean algo rithm which is taught in Sect However there is an easy way of telling whether an inverse for a given element a exists or not

An element a Z has a multiplicative inverse a 1 if and only if gcd a m 1 where gcd is the greatest common divisor i e the largest integer that divides

both numbers a and m The fact that two numbers have a gcd of 1 is of great importance in number theory and there is a special name for it if gcd a m 1 then a and m are said to be relatively prime or coprime

Example 0 Let s see whether the multiplicative inverse of 15 exists in Z26 Because

the inverse must exist On the other hand since

the multiplicative inverse of 14 does not exist in Z26

Another ring property is that a b c a b a c for all a b c Zm i e the distributive law holds In summary roughly speaking we can say that the ring Zm is the set of integers 0 1 2 m 1 in which we can add subtract multiply and sometimes divide

As mentioned earlier the ring Zm and thus integer arithmetic with the modulo operation is of central importance to modern public key cryptography In practice

the integers involved have a length of 96 bits so that efficient modular com putations are a crucial aspect

Shift Cipher or Caesar Cipher

We now introduce another historical cipher the shift cipher It is actually a special case of the substitution cipher and has a very elegant mathematical description

The shift cipher itself is extremely simple We simply shift every plaintext letter by a fixed number of positions in the alphabet For instance if we shift by 3 posi tions A would be substituted by d B by e etc The only problem arises towards the end of the alphabet what should we do with X Y Z As you might have guessed they should wrap around That means X should become a Y should be come b and Z is replaced by c Allegedly Julius Caesar used this cipher with a three position shift

The shift cipher also has an elegant description using modular arithmetic For the mathematical statement of the cipher the letters of the alphabet are encoded as numbers as depicted in Table

Encoding of letters for the shift cipher

Both the plaintext letters and the ciphertext letters are now elements of the ring Z26 Also the key i e the number of shift positions is also in Z26 since more than 26 shifts would not make sense 27 shifts would be the same as 1 shift etc The encryption and decryption of the shift cipher follows now as

Example 1 Let the key be k 17 and the plaintext is

ATTACK x1 x2 x6 0 19 19 0 2 10

The ciphertext is then computed as

y1 y2 y6 17 10 10 17 19 1 rkkrtb

As you can guess from the discussion of the substitution cipher earlier in this book the shift cipher is not secure at all There are two ways of attacking it

Since there are only 26 different keys shift positions one can easily launch a brute force attack by trying to decrypt a given ciphertext with all possible 26 keys If the resulting plaintext is readable text you have found the key

As for the substitution cipher one can also use letter frequency analysis

Now we try to improve the shift cipher by generalizing the encryption function Recall that the actual encryption of the shift cipher was the addition of the key yi xi k mod 26 The affine cipher encrypts by multiplying the plaintext by one part of the key followed by addition of another part of the key

The decryption is easily derived from the encryption function

x a 1 y b mod 26

The restriction gcd a 26 1 stems from the fact that the key parameter a needs to be inverted for decryption We recall from Sect that an element a and the modulus must be relatively prime for the inverse of a to exist Thus a must be in the set

a 1 3 5 7 9 11 15 17 19 21 23 25

But how do we find a 1 For now we can simply compute it by trial and error For a given a we simply try all possible values a 1 until we obtain

For instance if a 3 then a 1 9 since 27 1 mod 26 Note that a 1 also always fulfills the condition gcd a 1 26 1 since the inverse of a 1 always exists In fact the inverse of a 1 is a itself Hence for the trial and error determination of a 1 one only has to check the values given in Eq

Example 2 Let the key be k a b 9 13 and the plaintext be

ATTACK x1 x2 x6 0 19 19 0 2 10

The inverse a 1 of a exists and is given by a 1 3 The ciphertext is computed as

y1 y2 y6 13 2 2 13 5 25 nccnfz

Is the affine cipher secure No The key space is only a bit larger than in the case

key space values for a values for b

A key space with elements can of course still be searched exhaustively i e brute force attacked in a fraction of a second with current desktop PCs In addition the affine cipher has the same weakness as the shift and substitution cipher The mapping between plaintext letters and ciphertext letters is fixed Hence it can easily be broken with letter frequency analysis

The remainder of this book deals with strong cryptographic algorithms which are of practical relevance

Discussion and Further Reading

This book addresses practical aspects of cryptography and data security and is in tended to be used as an introduction it is suited for classroom use distance learning and self study At the end of each chapter we provide a discussion section in which we briefly describe topics for readers interested in further study of the material

About This Chapter Historical Ciphers and Modular Arithmetic This chapter introduced a few historical ciphers However there are many many more ranging from ciphers in ancient times to WW II encryption methods To readers who wish to learn more about historical ciphers and the role they played over the centuries the books by Bauer 13 Kahn 97 and Singh are highly recommended Besides making fascinating bedtime reading these books help one to understand the role that military and diplomatic intelligence played in shaping world history They also help to show modern cryptography in a larger context

The mathematics introduced in this chapter modular arithmetic belongs to the field of number theory This is a fascinating subject area which is unfortunately historically viewed as a branch of mathematics without applications Thus it is rarely taught outside mathematics curricula There is a wealth of books on number theory Among the classic introductory books are references A particu larly accessible book written for non mathematications is

Discussion and Further Reading 21

Research Community and General References Even though cryptography has matured considerably over the last 30 years it is still a relatively young field com pared to other disciplines and every year brings many new developments and dis coveries Many research results are published at events organized by the Interna tional Association for Cryptologic Research IACR The proceedings of the three IACR conferences CRYPTO EUROCRYPT and ASIACRYPT as well as the IACR workshops Cryptographic Hardware and Embedded Systems CHES Fast Soft ware Encryption FSE Public Key Cryptography PKC and Theory of Cryp tograpy TCC are excellent sources for tracking the recent developments in the field of cryptology at large Two important conferences which deal with the larger issue of security of which cryptography is one aspect are the IEEE Symposium on Security and Privacy and the USENIX Security forum All of the events listed take place annually

There are several good books on cryptography As reference sources the Hand book of Applied Cryptography and the more recent Encyclopedia of Cryptog raphy and Security are highly recommended both make excellent additions to this textbook

Provable Security Due to our focus on practical cryptography this book omits most aspects related to the theoretical foundations of crypto algorithms and proto cols Especially in modern cryptographic research there is a strong desire to provide statements about cryptographic schemes which are provable in a strict mathematical sense For this the goals of both a security system and the adversary are described in a formal model Often proofs are achieved by reducing the security of a system to certain assumptions e g that factorization of integers is hard or that a hash function is collision free

The field of provable security is quite large We list now some important subareas A recent survey on the specific area of provable public key encryption is given in 55 Provable security is closely related to cryptographic foundations which stud ies the general assumptions and approaches needed For instance the interrelation ship between certain presumably hard problems e g integer factorization and dis crete logarithm are studied The standard references are 81 83 Zero knowledge proofs are concerned with proving a certain knowledge towards another party with out revealing the secret They were originally motivated by proving an entity s iden tity without revealing a password or key However they are typically not used that way any more An early reference is and a more recent tutorial is given in 82 Multiparty computation can be used to compute answers such as the outcome of an election or determining the highest bid in an auction based on encrypted data The interesting part is that when the protocol is completed the participants know only their own input and the answer but nothing about the encrypted data of the other participants Good reference sources are and 83

A few times this book also touches upon provable security for instance the re lationship between Diffie Hellman key exchange and the Diffie Hellman problem cf Sect the block cipher based hash functions in Sect 2 or the security of the HMAC message authentication scheme in Sect

As a word of caution it should be mentioned that even though very practical results have been derived from research in the provable security of crypto schemes many findings are only of limited practical value Also the whole field is not without controversy 84

Secure System Design Cryptography is often an important tool for building a se cure system but on the other hand secure system design encompasses many other aspects Security systems are intended to protect something valuable e g informa tion monetary values personal property etc The main objective of secure system design is to make breaking the system more costly than the value of the protected assets where the cost should be measured in monetary value but also in more abstract terms such as effort or reputation Generally speaking adding security to a system often narrows its usability

In order to approach the problem systematically several general frameworks ex ist They typically require that assets and corresponding security needs have to be defined and that the attack potential and possible attack paths must be evaluated Finally adequate countermeasures have to be specified in order to realize an appro priate level of security for a particular application or environment

There are standards which can be used for evaluation and help to define a se cure system Among the more prominent ones are ISO IEC 94 8 6 0 1 2 7 the Common Criteria for Information Tech nology Security Evaluation 46 the German IT Grundschutzhandbuch 37 FIPS PUBS 77 and many more

Never ever develop your own crypto algorithm unless you have a team of expe rienced cryptanalysts checking your design

Do not use unproven crypto algorithms i e symmetric ciphers asymmetric ci phers hash functions or unproven protocols

Attackers always look for the weakest point of a cryptosystem For instance a large key space by itself is no guarantee for a cipher being secure the cipher might still be vulnerable against analytical attacks

Key lengths for symmetric algorithms in order to thwart exhaustive key search attacks are

64 bits insecure except for data with extremely short term value

8 bits long term security of several decades including attacks by in telligence agencies unless they possess quantum computers Based on our cur rent knowledge attacks are only feasible with quantum computers which do not exist and perhaps never will

bit as above but possibly against attacks by quantum computers

Modular arithmetic is a tool for expressing historical encryption schemes such as the affine cipher in a mathematically elegant way

The ciphertext below was encrypted using a substitution cipher Decrypt the ci phertext without knowledge of the key

lrvmnir bpr sumvbwvr jx bpr lmiwv yjeryrkbi jx qmbm wi bpr xjvni mkd ymibrut jx irhx wi bpr riirkvr jx ymbinlmtmipw utn qmumbr dj w ipmhh but bj rhnvwdmbr bpr yjeryrkbi jx bpr qmbm mvvjudwko bj yt wkbrusurbmbwjk lmird jk xjubt trmui jx ibndt

wb wi kjb mk rmit bmiq bj rashmwk rmvp yjeryrkb mkd wbi iwokwxwvmkvr mkd ijyr ynib urymwk nkrashmwkrd bj ower m vjyshrbr rashmkmbwjk jkr cjnhd pmer bj lr fnmhwxwrd mkd wkiswurd bj invp mk rabrkb bpmb pr vjnhd urmvp bpr ibmbr jx rkhwopbrkrd ywkd vmsmlhr jx urvjokwgwko ijnkdhrii ijnkd mkd ipmsrhrii ipmsr w dj kjb drry ytirhx bpr xwkmh mnbpjuwbt lnb yt rasruwrkvr cwbp qmbm pmi hrxb kj djnlb bpmb bpr xjhhjcwko wi bpr sujsru msshwvmbwjk mkd wkbrusurbmbwjk w jxxru yt bprjuwri wk bpr pjsr bpmb bpr riirkvr jx jqwkmcmk qmumbr cwhh urymwk wkbmvb

Compute the relative frequency of all letters A Z in the ciphertext You may want to use a tool such as the open source program CrypTool 50 for this task However a paper and pencil approach is also still doable

Decrypt the ciphertext with the help of the relative letter frequency of the English language see Table in Sect Note that the text is relatively short and that the letter frequencies in it might not perfectly align with that of general English language from the table

We received the following ciphertext which was encoded with a shift cipher

xultpaajcxitltlxaarpjhtiwtgxktghidhipxciwtvgtpilpit ghlxiwiwtxgqadds

Perform an attack against the cipher based on a letter frequency count How many letters do you have to identify through a frequency count to recover the key What is the cleartext

Who wrote this message

We consider the long term security of the Advanced Encryption Standard AES with a key length of bit with respect to exhaustive key search attacks AES is perhaps the most widely used symmetric cipher at this time

Assume that an attacker has a special purpose application specific integrated cir cuit ASIC which checks 08 keys per second and she has a budget of 1 million One ASIC costs 50 and we assume overhead for integrating

the ASIC manufacturing the printed circuit boards power supply cooling etc How many ASICs can we run in parallel with the given budget How long does an average key search take Relate this time to the age of the Universe which is about years

We try now to take advances in computer technology into account Predicting the future tends to be tricky but the estimate usually applied is Moore s Law which states that the computer power doubles every 18 months while the costs of integrated circuits stay constant How many years do we have to wait until a key search machine can be built for breaking AES with bit with an average search time of 24 hours Again assume a budget of 1 million do not take inflation into account

We now consider the relation between passwords and key size For this purpose we consider a cryptosystem where the user enters a key in the form of a password

Assume a password consisting of 8 letters where each letter is encoded by the ASCII scheme 7 bits per character i e possible characters What is the size of the key space which can be constructed by such passwords

What is the corresponding key length in bits

Assume that most users use only the 26 lowercase letters from the alphabet in stead of the full 7 bits of the ASCII encoding What is the corresponding key length in bits in this case

At least how many characters are required for a password in order to generate a key length of bits in case of letters consisting of

26 lowercase letters from the alphabet

As we learned in this chapter modular arithmetic is the basis of many cryp tosystems As a consequence we will address this topic with several problems in this and upcoming chapters

Let s start with an easy one Compute the result without a calculator

The results should be given in the range from 0 1 modulus 1 Briefly describe the relation between the different parts of the problem

Compute without a calculator

We consider the ring Z4 Construct a table which describes the addition of all elements in the ring with each other

Construct the multiplication table for Z4

Construct the addition and multiplication tables for Z5

Construct the addition and multiplication tables for Z6

There are elements in Z4 and Z6 without a multiplicative inverse Which ele ments are these Why does a multiplicative inverse exist for all nonzero elements in Z5

What is the multiplicative inverse of 5 in Z11 Z12 and Z13 You can do a trial and error search using a calculator or a PC

With this simple problem we want now to stress the fact that the inverse of an integer in a given ring depends completely on the ring considered That is if the modulus changes the inverse changes Hence it doesn t make sense to talk about an inverse of an element unless it is clear what the modulus is This fact is crucial for the RSA cryptosystem which is introduced in The extended Euclidean algorithm which can be used for computing inverses efficiently is introduced in Sect

Compute x as far as possible without a calculator Where appropriate make use of a smart decomposition of the exponent as shown in the example in Sect

The last problem is called a discrete logarithm and points to a hard problem which we discuss in The security of many public key schemes is based on the hardness of solving the discrete logarithm for large numbers e g with more than bits

Find all integers n between 0 n m that are relatively prime to m for m

4 5 9 26 We denote the number of integers n which fulfill the condition by m

e g 3 2 This function is called Euler s phi function What is m for m

This problem deals with the affine cipher with the key parameters a 7 b

Decrypt the text below

falszztysyjzyjkywjrztyjztyynaryjkyswarztyegyyj

Now we want to extend the affine cipher from Sect such that we can encrypt and decrypt messages written with the full German alphabet The German alphabet consists of the English one together with the three umlauts A O U and the even stranger double s character We use the following mapping from letters to integers

A 0 B 1 C 2 D 3 E 4 F 5

G 6 H 7 I 8 J 9 K 10 L 11

M 12 N 13 O 14 P 15 Q 16 R 17

S 18 T 19 U 20 V 21 W 22 X 23

Y 24 Z 25 A 26 O 27 U 28 29

What are the encryption and decryption equations for the cipher

How large is the key space of the affine cipher for this alphabet

The following ciphertext was encrypted using the key a 17 b 1 What is the corresponding plaintext

From which village does the plaintext come

In an attack scenario we assume that the attacker Oscar manages somehow to provide Alice with a few pieces of plaintext that she encrypts Show how Oscar can break the affine cipher by using two pairs of plaintext ciphertext x1 y1 and x2 y2 What is the condition for choosing x1 and x2

Remark In practice such an assumption turns out to be valid for certain settings e g encryption by Web servers etc This attack scenario is thus very important and is denoted as a chosen plaintext attack

An obvious approach to increase the security of a symmetric algorithm is to apply the same cipher twice i e

As is often the case in cryptography things are very tricky and results are often dif ferent from the expected and or desired ones In this problem we show that a double encryption with the affine cipher is only as secure as single encryption Assume two affine ciphers ek1 a1x b1 and ek2 a2x b2

Show that there is a single affine cipher ek3 a3x b3 which performs exactly the same encryption and decryption as the combination ek2 ek1 x

Find the values for a3 b3 when a1 3 b1 5 and a2 11 b2 7

For verification 1 encrypt the letter K first with ek1 and the result with ek2 and

2 encrypt the letter K with ek3

Briefly describe what happens if an exhaustive key search attack is applied to a double encrypted affine ciphertext Is the effective key space increased

Remark The issue of multiple encryption is of great practical importance in the case of the Data Encryption Standard DES for which multiple encryption in par ticular triple encryption does increase security considerably

If we look at the types of cryptographic algorithms that exist in a little bit more detail we see that the symmetric ciphers can be divided into stream ciphers and block ciphers as shown in Fig

Main areas within cryptography

This chapter gives an introduction to stream ciphers

The pros and cons of stream ciphers

Random and pseudorandom number generators

A truly unbreakable cipher the One Time Pad OTP

Linear feedback shift registers and Trivium a modern stream cipher

C Paar J Pelzl Understanding Cryptography 29

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

Stream Ciphers vs Block Ciphers

Symmetric cryptography is split into block ciphers and stream ciphers which are easy to distinguish Figure depicts the operational differences between stream Fig 2a and block Fig 2b ciphers when we want to encrypt b bits at a time where b is the width of the block cipher

xO x1 xb yO y1 yb xb yb

Principles of encrypting b bits with a stream a and a block b cipher

A description of the principles of the two types of symmetric ciphers follows

Stream ciphers encrypt bits individually This is achieved by adding a bit from a key stream to a plaintext bit There are synchronous stream ciphers where the key stream depends only on the key and asynchronous ones where the key stream also depends on the ciphertext If the dotted line in Fig is present the stream cipher is an asynchronous one Most practical stream ciphers are syn chronous ones and Sect of this chapter will deal with them An example of an asynchronous stream cipher is the cipher feedback CFB mode introduced in Sect 4

Synchronous and asynchronous stream ciphers

Block ciphers encrypt an entire block of plaintext bits at a time with the same key This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block In practice the vast majority of block ciphers either have a block length of bits 16 bytes such as the ad vanced encryption standard AES or a block length of 64 bits 8 bytes such as

the data encryption standard DES or triple DES 3DES algorithm All of these ciphers are introduced in later chapters

This chapter gives an introduction to stream ciphers Before we go into more detail it will be helpful to learn some useful facts about stream ciphers vs block ciphers

In practice in particular for encrypting computer communication on the Internet block ciphers are used more often than stream ciphers

Because stream ciphers tend to be small and fast they are particularly relevant for applications with little computational resources e g for cell phones or other small embedded devices A prominent example for a stream cipher is the A cipher which is part of the GSM mobile phone standard and is used for voice encryption However stream ciphers are sometimes also used for encrypting In ternet traffic especially the stream cipher RC4

Traditionally it was assumed that stream ciphers tended to encrypt more effi ciently than block ciphers Efficient for software optimized stream ciphers means that they need fewer processor instructions or processor cycles to encrypt one bit of plaintext For hardware optimized stream ciphers efficient means they need fewer gates or smaller chip area than a block cipher for encrypting at the same data rate However modern block ciphers such as AES are also very efficient in software Moreover for hardware there are also highly efficient block ciphers such as PRESENT which are as efficient as very compact stream ciphers

Encryption and Decryption with Stream Ciphers

As mentioned above stream ciphers encrypt plaintext bits individually The question now is How does encryption of an individual bit work The answer is surprisingly simple Each bit xi is encrypted by adding a secret key stream bit si modulo 2

Since encryption and decryption functions are both simple additions modulo 2 we can depict the basic operation of a stream cipher as shown in Fig Note that we use a circle with an addition sign as the symbol for modulo 2 addition

Just looking at the formulae there are three points about the stream cipher en cryption and decryption function which we should clarify

Encryption and decryption are the same functions

Encryption and decryption with stream ciphers

Why can we use a simple modulo 2 addition as encryption

What is the nature of the key stream bits si

The following discussion of these three items will give us already an understanding of some important stream cipher properties

Why Are Encryption and Decryption the Same Function

The reason for the similarity of the encryption and decryption function can easily be shown We must prove that the decryption function actually produces the plain text bit xi again We know that ciphertext bit yi was computed using the encryption function yi xi si mod 2 We insert this encryption expression in the decryption function

dsi yi yi si mod 2

xi si si mod 2

The trick here is that the expression 2 si mod 2 has always the value zero since

0 mod 2 Another way of understanding this is as follows If si has either the value 0 in which case 2 si 0 mod 2 If si 1 we have 2 si 2

Why Is Modulo 2 Addition a Good Encryption Function

A mathematical explanation for this is given in the context of the One Time Pad in Sect However it is worth having a closer look at addition modulo 2 If we do arithmetic modulo 2 the only possible values are 0 and 1 because if you divide by 2 the only possible remainders are 0 and 1 Thus we can treat arithmetic modulo 2 as Boolean functions such as AND gates OR gates NAND gates etc Let s look at the truth table for modulo 2 addition

This should look familiar to most readers It is the truth table of the exclusive OR also called XOR gate This is in important fact Modulo 2 addition is equivalent to

the XOR operation The XOR operation plays a major role in modern cryptography and will be used many times in the remainder of this book

The question now is why is the XOR operation so useful as opposed to for instance the AND operation Let s assume we want to encrypt the plaintext bit xi 0 If we look at the truth table we find that we are on either the 1st or 2nd line of the truth table

Truth table of the XOR operation

Depending on the key bit the ciphertext yi is either a zero si 0 or one si 1 If the key bit si behaves perfectly randomly i e it is unpredictable and has exactly a 50 chance to have the value 0 or 1 then both possible ciphertexts also occur with a 50 likelihood Likewise if we encrypt the plaintext bit xi 1 we are on line 3 or 4 of the truth table Again depending on the value of the key stream bit si there is a 50 chance that the ciphertext is either a 1 or a 0

We just observed that the XOR function is perfectly balanced i e by observing an output value there is exactly a 50 chance for any value of the input bits This distinguishes the XOR gate from other Boolean functions such as the OR AND or NAND gate Moreover AND and NAND gates are not invertible Let s look at a very simple example for stream cipher encryption

Example Alice wants to encrypt the letter A where the letter is given in ASCII code The ASCII value for A is Let s furthermore assume that the first key stream bits are s0 s6

Note that the encryption by Alice turns the uppercase A into the lower case letter

m Oscar the attacker who eavesdrops on the channel only sees the ciphertext letter

m Decryption by Bob with the same key stream reproduces the plaintext A again

So far stream ciphers look unbelievably easy One simply takes the plaintext performs an XOR operation with the key and obtains the ciphertext On the receiving side Bob does the same The only thing left to discuss is the last question from above

What Exactly Is the Nature of the Key Stream

It turns out that the generation of the values si which are called the key stream is the central issue for the security of stream ciphers In fact the security of a stream cipher completely depends on the key stream The key stream bits si are not the key bits themselves So how do we get the key stream Generating the key stream is pretty much what stream ciphers are about This is a major topic and is discussed later in this chapter However we can already guess that a central requirement for the key stream bits should be that they appear like a random sequence to an attacker Otherwise an attacker Oscar could guess the bits and do the decryption by himself Hence we first need to learn more about random numbers

Historical Remark Stream ciphers were invented in by Gilbert Vernam even though they were not called stream ciphers back at that time He built an elec tromechanical machine which automatically encrypted teletypewriter communica tion The plaintext was fed into the machine as one paper tape and the key stream as a second tape This was the first time that encryption and transmission was au tomated in one machine Vernam studied electrical engineering at Worcester Poly technic Institute WPI in Massachusetts where by coincidence one of the authors of this book was a professor in the s Stream ciphers are sometimes referred to as Vernam ciphers Occasionally one time pads are also called Vernam ciphers For further reading on Vernam s machine the book by Kahn 97 is recommended

Random Numbers and an Unbreakable Stream Cipher

Random Number Generators

As we saw in the previous section the actual encryption and decryption of stream ciphers is extremely simple The security of stream ciphers hinges entirely on a suitable key stream s0 s1 s2 Since randomness plays a major role we will first

learn about the three types of random number generators RNG that are important for us

True Random Number Generators TRNG

True random number generators TRNGs are characterized by the fact that their output cannot be reproduced For instance if we flip a coin times and record the resulting sequence of bits

to generate the same bit sequence The chance of success is 1 which is

an extremely small probability TRNGs are based on physical processes Examples include coin flipping rolling of dice semiconductor noise clock jitter in digital circuits and radioactive decay In cryptography TRNGs are often needed for gener ating session keys which are then distributed between Alice and Bob and for other purposes

General Pseudorandom Number Generators PRNG

Pseudorandom number generators PRNGs generate sequences which are com puted from an initial seed value Often they are computed recursively in the follow ing way

si 1 f si i 0 1

A generalization of this are generators of the form si 1 f si si 1 si t where

t is a fixed integer A

si 1 asi b mod m i 0 1

where a b m are integer constants Note that PRNGs are not random in a true sense because they can be computed and are thus completely deterministic A widely used example is the rand function used in ANSI C It has the parameters

si 1 45 si 5 mod i 0 1

A common requirement of PRNGs is that they possess good statistical proper ties meaning their output approximates a sequence of true random numbers There are many mathematical tests e g the chi square test which can verify the statistical behavior of PRNG sequences Note that there are many many applications for pseu dorandom numbers outside cryptography For instance many types of simulations or testing e g of software or of VLSI chips need random data as input That is the reason why a PRNG is included in the ANSI C specification

Cryptographically Secure Pseudorandom Number Generators CSPRNG

Cryptographically secure pseudorandom number generators CSPRNGs are a spe cial type of PRNG which possess the following additional property A CSPRNG is

PRNG which is unpredictable Informally this means that given n output bits of the key stream si si 1 si n 1 where n is some integer it is computationally infea sible to compute the subsequent bits si n si n 1 A more exact definition is that

given n consecutive bits of the key stream there is no polynomial time algorithm that can predict the next bit sn 1 with better than 50 chance of success Another property of CSPRNG is that given the above sequence it should be computationally infeasible to compute any preceding bits si 1 si 2

Note that the need for unpredictability of CSPRNGs is unique to cryptography In virtually all other situations where pseudorandom numbers are needed in com puter science or engineering unpredictability is not needed As a consequence the distinction between PRNG and CSPRN and their relevance for stream ciphers is of ten not clear to non cryptographers Almost all PRNG that were designed without the clear purpose of being stream ciphers are not CSPRNGs

In the following we discuss what happens if we use the three types of random num bers as generators for the key stream sequence s0 s1 s2 of a stream cipher Let s

first define what a perfect cipher should be

Unconditional security is based on information theory and assumes no limit on the attacker s computational power This looks like a pretty straightforward defini tion It is in fact straightforward but the requirements for a cipher to be uncondi tionally secure are tremendous Let s look at it using a gedankenexperiment As sume we have a symmetric encryption algorithm it doesn t matter whether it s a block cipher or stream cipher with a key length of 00 bits and the only attack that works is an exhaustive key search i e a brute force attack From the discussion in Sect 2 we recall that bits are more than enough for long term security So is a cipher with 00 bits unconditionally secure The answer is simple No Since an attacker can have infinite computational resources we can simply assume that the attacker has 00 computers available and every computer checks exactly one key This will give us a correct key in one time step Of course there is no way that 00 computer can ever be built the number is too large It is estimated that

there are only about atoms in the known universe The cipher would merely be computationally secure but not unconditionally

All this said we now show a way to build an unconditionally secure cipher that is quite simple This cipher is called the One Time Pad

It is easy to show why the OTP is unconditionally secure Here is a sketch of a proof For every ciphertext bit we get an equation of this form

Each individual relation is a linear equation modulo 2 with two unknowns They are impossible to solve If the attacker knows the value for y0 0 or 1 he cannot determine the value of x0 In fact the solutions x0 0 and x0 1 are exactly equally likely if s0 stems from a truly random source and there is 50 chance that it has the value 0 and 1 The situation is identical for the second equation and all subsequent ones Note that the situation is different if the values si are not truly random In this case there is some functional relationship between them and the equations shown above are not independent Even though it might still be hard to solve the system of equations it is not provably secure

So now we have a simple cipher which is perfectly secure There are rumors that the red telephone between the White House and the Kremlin was encrypted using an OTP during the Cold War Obviously there must be a catch since OTPs are not used for Web browsers e mail encryption smart cards mobile phones or other important applications Let s look at the implications of the three requirements in Defintion 2 The first requirement means that we need a TRNG That means we need a device e g based on white noise of a semiconductor that generates truly random bits Since standard PCs do not have TRNG this requirement might not be that convenient but can certainly be met The second requirement means that Alice has to get the random bits securely to Bob In practice that could mean that Alice burns the true random bits on a CD ROM and sends them securely e g with a trusted courier to Bob Still doable but not great The third requirement is probably

the most impractical one Key stream bits cannot be re used This implies that we need one key bit for every bit of plaintext Hence our key is as long as the plaintext This is probably the major drawback of the OTP Even if Alice and Bob share a CD with 1 MByte of true random numbers we run quickly into limits If they send a single email with an attachment of 1 MByte they could encrypt and decrypt it but after that they would need to exchange a true random key stream again

For these reasons OTPs are rarely used in practice However they give us a great design idea for secure ciphers If we XOR truly random bits and plaintext we get ciphertext that can certainly not be broken by an attacker We will see in the next section how we can use this fact to build practical stream ciphers

Towards Practical Stream Ciphers

In the previous section we saw that OTPs are unconditionally secure but that they have drawbacks which make them impractical What we try to do with practical stream ciphers is to replace the truly random key stream bits by a pseudorandom number generator where the key k serves as a seed The principle of practical stream ciphers is shown in Fig

Practical stream ciphers

Before we turn to stream ciphers used in the real world it should be stressed that practical stream ciphers are not unconditionally secure In fact all known practical crypto algorithms stream ciphers block ciphers public key algorithms are not unconditionally secure The best we can hope for is computational security which we define as follows

This seems like a reasonable definition but there are still several problems with it First often we do not know what the best algorithm for a given attack is A prime example is the RSA public key scheme which can be broken by factoring large integers Even though many factoring algorithms are known we do not know whether there exist any better ones Second even if a lower bound on the complexity of one attack is known we do not know whether any other more powerful attacks are possible We saw this in Sect 2 during the discussion about the substitution cipher Even though we know the exact computational complexity for an exhaustive key search there exist other more powerful attacks The best we can do in practice is to design crypto schemes for which it is assumed that they are computationally secure For symmetric ciphers this usually means one hopes that there is no attack method with a complexity better than an exhaustive key search

Let s go back to Fig This design emulates behaves like a one time pad It has the major advantage over the OTP that Alice and Bob only need to exchange a secret key that is at most a few bits long and that does not have to be as long as the message we want to encrypt We now have to think carefully about the properties of the key stream s0 s1 s2 that is generated by Alice and Bob Obviously we need some type of random number generator to derive the key stream First we note that we cannot use a TRNG since by definition Alice and Bob will not be able to generate the same key stream Instead we need deterministic i e pseudorandom number generators We now look at the other two generators that were introduced in the previous section

Building Key Streams from PRNGs

Here is an idea that seems promising but in fact is pretty bad Many PRNGs pos sess good statistical properties which are necessary for a strong stream cipher If we apply statistical tests to the key stream sequence the output should pretty much behave like the bit sequence generated by tossing a coin So it is tempting to assume that a PRNG can be used to generate the key stream But all of this is not sufficient for a stream cipher since our opponent Oscar is smart Consider the following at tack

Example Let s assume a PRNG based on the linear congruential generator

Si 1 A Si B mod m i 0 1

where we choose m to be bits long and Si A B 0 1 m 1 Note that this PRNG can have excellent statistical properties if we choose the parameters carefully The modulus m is part of the encryption scheme and is publicly known The secret key comprises the values A B and possibly the seed S0 each with a length of That gives us a key length of bit which is more than sufficient to protect against a brute force attack Since this is a stream cipher Alice can encrypt

where si are the bits of the binary representation of the PRNG output symbols S j

But Oscar can easily launch an attack Assume he knows the first bits of plaintext this is only 3 byte e g file header information or he guesses part of the plaintext Since he certainly knows the ciphertext he can now compute the first bits of key stream as

si yi xi mod m i 1 2

These bits immediately give the first three output symbols of the PRNG S1 s1 s S2 s s and S3 s s Oscar can now generate two equations

S2 A S1 B mod m S3 A S2 B mod m

This is a system of linear equations over Zm with two unknowns A and B But those two values are the key and we can immediately solve the system yielding

A S2 S3 S1 S2 mod m

B S2 S1 S2 S3 S1 S2 mod m

In case gcd S1 S2 m 1 we get multiple solutions since this is an equation sys tem over Zm However with a fourth piece of known plaintext the key can uniquely be detected in almost all cases Alternatively Oscar simply tries to encrypt the mes sage with each of the multiple solutions found Hence in summary if we know a few pieces of plaintext we can compute the key and decrypt the entire ciphertext

This type of attack is why the notation of CSPRNG was invented

Building Key Streams from CSPRNGs

What we need to do to prevent the attack above is to use a CSPRNG which assures that the key stream is unpredictable We recall that this means that given the first n output bits of the key stream s1 s2 sn it is computationally infeasible to com pute the bits sn 1 sn 2 Unfortunately pretty much all pseudorandom number

generators that are used for applications outside cryptography are not cryptograph ically secure Hence in practice we need to use specially designed pseudorandom number generators for stream ciphers

The question now is how practical stream ciphers actually look There are many proposals for stream ciphers out in the literature They can roughly be classified as ciphers either optimized for software implementation or optimized for hardware im plementation In the former case the ciphers typically require few CPU instructions

to compute one key stream bit In the latter case they tend to be based on operations which can easily be realized in hardware A popular example is shift registers with feedback which are discussed in the next section A third class of stream ciphers is realized by using block ciphers as building blocks The cipher feedback mode output feedback mode and counter mode to be introduced in are examples of stream ciphers derived from block ciphers

It could be argued that the state of the art in block cipher design is more ad vanced than stream ciphers Currently it seems to be easier for scientists to design secure block ciphers than stream ciphers Subsequent chapters deal in great detail with the two most popular and standardized block ciphers DES and AES

Shift Register Based Stream Ciphers

As we have learned so far practical stream ciphers use a stream of key bits s1 s2 that are generated by the key stream generator which should have certain properties An elegant way of realizing long pseudorandom sequences is to use linear feedback shift registers LFSRs LFSRs are easily implemented in hardware and many but certainly not all stream ciphers make use of LFSRs A prominent example is the A cipher which is standardized for voice encryption in GSM As we will see even though a plain LFSR produces a sequence with good statistical properties it is cryptographically weak However combinations of LFSRs such as A or the cipher Trivium can make secure stream ciphers It should be stressed that there are many ways for constructing stream ciphers This section only introduces one of several popular approaches

Linear Feedback Shift Registers LFSR

An LFSR consists of clocked storage elements flip flops and a feedback path The number of storage elements gives us the degree of the LFSR In other words an LFSR with m flip flops is said to be of degree m The feedback network computes the input for the last flip flop as XOR sum of certain flip flops in the shift register

Example Simple LFSR We consider an LFSR of degree m 3 with flip flops FF2 FF1 FF0 and a feedback path as shown in Fig The internal state bits are denoted by si and are shifted by one to the right with each clock tick The rightmost state bit is also the current output bit The leftmost state bit is computed in the feedback path which is the XOR sum of some of the flip flop values in the previous clock period Since the XOR is a linear operation such circuits are called linear feedback shift registers If we assume an initial state of s2 1 s1 0 s0 0 Table gives the complete sequence of states of the LFSR Note that the rightmost column is the output of the LFSR One can see from this example that the LFSR

Linear feedback shift register of degree 3 with initial values s2 s1 s0

Sequence of states of the LFSR

starts to repeat after clock cycle 6 This means the LFSR output has period of length 7 and has the form

There is a simple formula which determines the functioning of this LFSR Let s look at how the output bits si are computed assuming the initial state bits s0 s1 s2

s3 s1 s0 mod 2 s4 s2 s1 mod 2 s5 s3 s2 mod 2

In general the output bit is computed as

si 3 si 1 si mod 2

where i 0 1 2

This was of course a simple example However we could already observe many important properties We will now look at general LFSRs

A Mathematical Description of LFSRs

The general form of an LFSR of degree m is shown in Fig It shows m flip flops and m possible feedback locations all combined by the XOR operation Whether a feedback path is active or not is defined by the feedback coefficient p0 p1 pm 1

If pi 1 closed switch the feedback is active

If pi 0 open switch the corresponding flip flop output is not used for the feedback

With this notation we obtain an elegant mathematical description for the feedback path If we multiply the output of flip flop i by its coefficient pi the result is either the output value if pi 1 which corresponds to a closed switch or the value zero if pi 0 which corresponds to an open switch The values of the feedback coefficients are crucial for the output sequence produced by the LFSR

General LFSR with feedback coefficients pi and initial values sm 1 s0

Let s assume the LFSR is initially loaded with the values s0 sm 1 The next output bit of the LFSR sm which is also the input to the leftmost flip flop can be

computed by the XOR sum of the products of flip flop outputs and corresponding feedback coefficient

sm sm 1 pm 1 s1 p1 s0 p0 mod 2 The next LFSR output can be computed as

sm 1 sm pm 1 s2 p1 s1 p0 mod 2 In general the output sequence can be described as

si m p j si j mod 2 si p j 0 1 i 0 1 2

Clearly the output values are given through a combination of some previous output values LFSRs are sometimes referred to as linear recurrences

Due to the finite number of recurring states the output sequence of an LFSR re peats periodically This was also illustrated in Example Moreover an LFSR can produce output sequences of different lengths depending on the feedback coeffi cients The following theorem gives us the maximum length of an LFSR as function of its degree

It is easy to show that this theorem holds The state of an LFSR is uniquely deter mined by the m internal register bits Given a certain state the LFSR deterministi cally assumes its next state Because of this as soon as an LFSR assumes a previous state it starts to repeat Since an m bit state vector can only assume 2m 1 nonzero states the maximum sequence length before repetition is 2m 1 Note that the all zero state must be excluded If an LFSR assumes this state it will get stuck in it i e it will never be able to leave it again Note that only certain configurations p0 pm 1 yield maximum length LFSRs We give a small example for this be low

Example LFSR with maximum length output sequence

Given an LFSR of degree m 4 and the feedback path p3 0 p2 0 p1 1 p0 1 the output sequence of the LFSR has a period of 2m 1 15 i e it is a maximum length LFSR

Example LFSR with non maximum output sequence

Given an LFSR of degree m 4 and p3 1 p2 1 p1 1 p0 1 then the output sequence has period of 5 therefore it is not a maximum length LFSR

The mathematical background of the properties of LFSR sequences is beyond the scope of this book However we conclude this introduction to LFSRs with some additional facts LFSRs are often specified by polynomials using the following no tation An LFSR with a feedback coefficient vector pm 1 p1 p0 is represented by the polynomial

P x xm pm 1xm 1 p1x p0

For instance the LFSR from the example above with coefficients p3 0 p2 0 p1 1 p0 1 can alternatively be specified by the polynomial x4 x 1 This seemingly odd notation as a polynomial has several advantages For instance maximum length LFSRs have what is called primitive polynomials Primitive poly nomials are a special type of irreducible polynomial Irreducible polynomials are roughly comparable with prime numbers i e their only factors are 1 and the polynomial itself Primitive polynomials can relatively easily be computed Hence maximum length LFSRs can easily be found Table shows one primitive poly nomial for every value of m in the range from m 2 3 As an example

the notation 0 2 5 refers to the polynomial 1 x2 x5 Note that there are many primitive polynomials for every given degree m For instance there exist 66 different primitive polynomials of degree m 31

Primitive polynomials for maximum length LFSRs

2 24 46 68 90

3 25 47 69 91

4 26 48 70 92

5 27 49 71 93

6 28 50 9 2 94

7 29 51 73 0 5

8 30 52 74 9 6

9 31 53 75 97

10 32 54 76 98

11 33 55 77 99

12 34 56 78

13 35 57 79

14 36 58 80 0 24

15 37 59 81

16 38 60 82

17 39 61 83

18 40 62 84

19 41 63 85

20 42 64 86

21 43 65 87

22 44 66 9 8

23 45 67 89

Known Plaintext Attack Against Single LFSRs

As indicated by its name LFSRs are linear Linear systems are governed by linear relationships between their inputs and outputs Since linear dependencies can rela tively easily be analyzed this can be a major advantage e g in communication sys tems However a cryptosystem where the key bits only occur in linear relationships makes a highly insecure cipher We will now investigate how the linear behavior of a LFSR leads to a powerful attack

If we use an LFSR as a stream cipher the secret key k is the feedback coefficient vector pm 1 p1 p0 An attack is possible if the attacker Oscar knows some plaintext and the corresponding ciphertext We further assume that Oscar knows the degree m of the LFSR The attack is so efficient that he can easily try a large num

ber of possible m values so that this assumption is not a major restriction Let the known plaintext be given by x0 x1 x2m 1 and the corresponding ciphertext by y0 y1 y2m 1 With these 2m pairs of plaintext and ciphertext bits Oscar recon

structs the first 2m key stream bits

si xi yi mod 2 i 0 1 2m 1

The goal is now to find the key which is given by the feedback coefficients pi

Eq is a description of the relationship of the unknown key bits pi and the key stream output We repeat the equation here for convenience

si m p j si j mod 2 si p j 0 1 i 0 1 2

Note that we get a different equation for every value of i Moreover the equations are linearly independent With this knowledge Oscar can generate m equations for the first m values of i

i 0 sm pm 1sm 1 p p mod 2

i 1 sm 1 pm 1sm p p mod 2

i m 1 s2m 1 pm m 2 p1sm p0sm 1 mod 2

He has now m linear equations in m unknowns p0 p1 pm 1 This system can easily be solved by Oscar using Gaussian elimination matrix inversion or any other algorithm for solving systems of linear equations Even for large values of m this can be done easily with a standard PC

This situation has major consequences as soon as Oscar knows 2m output bits of an LFSR of degree m the pi coefficients can exactly be constructed by merely solving a system of linear equations Once he has computed these feedback coef ficients he can build the LFSR and load it with any m consecutive output bits that he already knows Oscar can now clock the LFSR and produce the entire output sequence Because of this powerful attack LFSRs by themselves are extremely inse cure They are a good example of a PRNG with good statistical properties but with terrible cryptographical ones Nevertheless all is not lost There are many stream ciphers which use combinations of several LFSRs to build strong cryptosystems The cipher Trivium in the next section is an example

Trivium is a relatively new stream cipher which uses an 80 bit key It is based on a combination of three shift registers Even though these are feedback shift registers there are nonlinear components used to derive the output of each register unlike the LFSRs that we studied in the previous section

Description of Trivium

As shown in Fig at the heart of Trivium are three shift registers A B and C The lengths of the registers are 93 84 and respectively The XOR sum of all three register outputs forms the key stream si A specific feature of the cipher is that

Internal structure of the stream cipher Trivium

the output of each register is connected to the input of another register Thus the registers are arranged in circle like fashion The cipher can be viewed as consisting of one circular register with a total length of 93 84 Each of the three registers has similar structure as described below

The input of each register is computed as the XOR sum of two bits

The output bit of another register according to Fig For instance the output of register A is part of the input of register B

One register bit at a specific location is fed back to the input The positions are given in Table For instance bit 68 of register A is fed back to its input

The output of each register is computed as the XOR sum of three bits

The rightmost register bit

One register bit at a specific location is fed forward to the output The positions are given in Table For instance bit 66 of register A is fed to its output

The output of a logical AND function whose input is two specific register bits Again the positions of the AND gate inputs are given in Table

Specification of Trivium

Note that the AND operation is equal to multiplication in modulo 2 arithmetic If we multiply two unknowns and the register contents are the unknowns that an at tacker wants to recover the resulting equations are no longer linear as they contain products of two unknowns Thus the feedforward paths involving the AND opera tion are crucial for the security of Trivium as they prevent attacks that exploit the

linearity of the cipher as the one applicable to plain LFSRs shown in the previous section

Encryption with Trivium

Almost all modern stream ciphers have two input parameters a key k and an ini tialization vector IV The former is the regular key that is used in every symmetric crypto system The IV serves as a randomizer and should take a new value for every encryption session It is important to note that the IV does not have to be kept secret it merely must change for every session Such values are often referred to as nonces which stands for number used once Its main purpose is that two key streams pro duced by the cipher should be different even though the key has not changed If this were not the case the following attack becomes possible If an attacker has known plaintext from a first encryption he can compute the corresponding key stream The second encryption using the same key stream can now immediately be deciphered Without a changing IV stream cipher encryption is highly deterministic Methods for generating IVs are discussed in Sect 2 Let s look at the details of running Trivium

Initialization Initially an 80 bit IV is loaded into the 80 leftmost locations of reg ister A and an 80 bit key is loaded in the 80 leftmost locations of register B All other register bits are set to zero with the exception of the three rightmost bits of register C i e bits c c and c which are set to 1

Warm up Phase In the first phase the cipher is clocked 88 times No cipher output is generated

Encryption Phase The bits produced hereafter i e starting with the output bit of cycle form the key stream

The warm up phase is needed for randomizing the cipher sufficiently It makes sure that the key stream depends on both the key k and the IV

An attractive feature of Trivium is its compactness especially if implemented in hardware It mainly consists of a bit shift register and a few Boolean oper ations It is estimated that a hardware implementation of the cipher occupies and area of between about and gate equivalences depending on the degree of parallelization A gate equivalence is the chip area occupied by a 2 input NAND gate For instance an implementation with gates computes the key stream at a rate of 16 bits clock cycle This is considerably smaller than most block ciphers such as AES and is very fast If we assume that this hardware design is clocked at a moderate MHz the encryption rate would be 16bit MHz 2 Gbit sec In software it is estimated that computing 8 output bits takes 12 cycles on a GHz Intel CPU resulting in a theoretical encryption rate of 1 Gbit sec

Even though there are no known attacks at the time of writing one should keep in mind that Trivium is a relatively new cipher and attacks in the future are certainly

Discussion and Further Reading 49

a possibility In the past many other stream ciphers were found to be not secure More information on Trivium can be found in

Discussion and Further Reading

Established Stream Ciphers Even though many stream ciphers have been pro posed over the years there are considerably fewer well investigated ones The se curity of many proposed stream ciphers is unknown and many stream ciphers have been broken In the case of software oriented stream ciphers arguably the best investigated ones are RC4 and SEAL Sect 1 Note that there are some known weaknesses in RC4 even though it is still secure in practice if it is used correctly The SEAL cipher on the other hand is patented

In the case of hardware oriented ciphers there is a wealth of LFSR based al gorithms Many proposed ciphers have been broken see references 8 85 for an introduction Among the best studied ones are the A and A algorithms which are used in GSM mobile networks for voice encryption between cell phones and base stations A which is the cipher used in most industrialized nations had originally been kept secret but was reverse engineered and published on the Internet in The cipher is borderline secure today 22 whereas the weaker A has much more serious flaws 11 Neither of the two ciphers is recommended based on today s understanding of cryptanalysis For 3GPP mobile communication a differ ent cipher A also named KASUMI is used but it is a block cipher

This somewhat pessimistic outlook on the state of the art in stream ciphers changed with the eSTREAM project described below

eSTREAM Project The eSTREAM project had the explicit goal to advance the state of the art knowledge about stream cipher design As part of this objective new stream ciphers that might become suitable for widespread adoption were in vestigated eSTREAM was organized by the European Network of Excellence in Cryptography ECRYPT The call for stream ciphers was first issued in November and ended in The ciphers were divided into two profiles depending on the intended application

Profile 1 Stream ciphers for software applications with high throughput require ments

Profile 2 Stream ciphers for hardware applications with restricted resources such as limited storage gate count or power consumption

Some cryptographers had emphasized the importance of including an authentication method and hence two further profiles were also included to deal with ciphers that also provide authentication

A total of 34 candidates were submitted to eSTREAM At the end of the project four software oriented Profile 1 ciphers were found to have desirable properties HC Rabbit Salsa 2 and SOSEMANUK With respect to hardware oriented ciphers Profile 2 the following three ciphers were selected Grain v1 MICKEY

v2 and Trivium Note that all of these are relatively new ciphers and only time will tell whether they are really cryptographically strong The algorithm descrip tion source code and the results of the four year evaluation process are available online 69 and the official book provides more detailed information

It is important to keep in mind that ECRYPT is not a standardization body so the status of the eSTREAM finalist ciphers cannot be compared to that of AES at the end of its selection process cf Sect

True Random Number Generation We introduced in this chapter different classes of RNGs and found that cryptographically secure pseudorandom number genera tors are of central importance for stream ciphers For other cryptographic appli cations true random number generators are important For instance true random numbers are needed for the generation of cryptographic keys which are then to be distributed Many ciphers and modes of operation rely on initial values that are of ten generated from TRNGs Also many protocols require nonces numbers used only once which may stem from a TRNG All TRNGs need to exploit some en tropy source i e some process which behaves truly randomly Many TRNG designs have been proposed over the years They can coarsely be classified as approaches that use specially designed hardware as an entropy source or as TRNGs that exploit external sources of randomness Examples of the former are circuits with random behavior e g that are based on semiconductor noise or on several uncorrelated os cillators Reference contains a good survey Examples of the latter ones are computer systems which measure the times between key strokes or the arrival times of packets at a network interface In all these cases one has to be ex tremely careful to make sure that the noise source in fact has enough entropy There are many examples of TRNG designs which turned out to have poor random behav ior and which constitute a serious security weakness depending on how they are used There are tools available that test the statistical properties of TRNG output sequences 56 There are also standards with which TRNGs can be formally evaluated 80

Stream ciphers are less popular than block ciphers in most domains such as Inter net security There are exceptions for instance the popular stream cipher RC4

Stream ciphers sometimes require fewer resources e g code size or chip area for implementation than block ciphers and they are attractive for use in con strained environments such as cell phones

The requirements for a cryptographically secure pseudorandom number gener ator are far more demanding than the requirements for pseudorandom number generators used in other applications such as testing or simulation

The One Time Pad is a provable secure symmetric cipher However it is highly impractical for most applications because the key length has to equal the message length

Single LFSRs make poor stream ciphers despite their good statistical properties However careful combinations of several LFSR can yield strong ciphers

The stream cipher described in Definition 1 can easily be generalized to work in alphabets other than the binary one For manual encryption an especially useful one is a stream cipher that operates on letters

Develop a scheme which operates with the letters A B Z represented by the numbers 25 What does the key stream look like What are the encryp tion and decryption functions

Decrypt the following cipher text

which was encrypted using the key

How was the young man murdered

Assume we store a one time key on a CD ROM with a capacity of 1 Gbyte Discuss the real life implications of a One Time Pad OTP system Address issues such as life cycle of the key storage of the key during the life cycle after the life cycle key distribution generation of the key etc

Assume an OTP like encryption with a short key of bit This key is then being used periodically to encrypt large volumes of data Describe how an attack works that breaks this scheme

At first glance it seems as though an exhaustive key search is possible against an OTP system Given is a short message let s say 5 ASCII characters represented by 40 bit which was encrypted using a 40 bit OTP Explain exactly why an exhaus tive key search will not succeed even though sufficient computational resources are available This is a paradox since we know that the OTP is unconditionally secure That is explain why a brute force attack does not work

Note You have to resolve the paradox That means answers such as The OTP is unconditionally secure and therefore a brute force attack does not work are not valid

We will now analyze a pseudorandom number sequence generated by a LFSR characterized by c2 1 c1 0 c0 1

What is the sequence generated from the initialization vector s2 1 s1 0 s0

What is the sequence generated from the initialization vector s2 0 s1 1 s0

How are the two sequences related

Assume we have a stream cipher whose period is quite short We happen to know that the period is 0 bit in length We assume that we do not know anything else about the internals of the stream cipher In particular we should not assume that it is a simple LFSR For simplicity assume that English text in ASCII format is being encrypted

Describe in detail how such a cipher can be attacked Specify exactly what Oscar has to know in terms of plaintext ciphertext and how he can decrypt all ciphertext

Compute the first two output bytes of the LFSR of degree 8 and the feedback polynomial from Table where the initialization vector has the value FF in hex adecimal notation

In this problem we will study LFSRs in somewhat more detail LFSRs come in three flavors

LFSRs which generate a maximum length sequence These LFSRs are based on

primitive polynomials

LFSRs which do not generate a maximum length sequence but whose sequence length is independent of the initial value of the register These LFSRs are based on irreducible polynomials that are not primitive Note that all primitive polyno mials are also irreducible

LFSRs which do not generate a maximum length sequence and whose sequence length depends on the initial values of the register These LFSRs are based on reducible polynomials

We will study examples in the following Determine all sequences generated by

Draw the corresponding LFSR for each of the three polynomials Which of the polynomials is primitive which is only irreducible and which one is reducible Note that the lengths of all sequences generated by each of the LFSRs should add

Given is a stream cipher which uses a single LFSR as key stream generator The LFSR has a degree of

How many plaintext ciphertext bit pairs are needed to launch a successful attack

Describe all steps of the attack in detail and develop the formulae that need to be solved

What is the key in this system Why doesn t it make sense to use the initial contents of the LFSR as the key or as part of the key

We conduct a known plaintext attack on an LFSR based stream cipher We know that the plaintext sent was

By tapping the channel we observe the following stream

What is the degree m of the key stream generator

What is the initialization vector

Determine the feedback coefficients of the LFSR

Draw a circuit diagram and verify the output sequence of the LFSR

We want to perform an attack on another LFSR based stream cipher In order to process letters each of the 26 uppercase letters and the numbers 0 1 2 3 4 5 are represented by a 5 bit vector according to the following mapping

We happen to know the following facts about the system

The degree of the LFSR is m 6

Every message starts with the header WPI

We observe now on the channel the following message the fourth letter is a zero

What is the initialization vector

What are the feedback coefficients of the LFSR

Write a program in your favorite programming language which generates the whole sequence and find the whole plaintext

Where does the thing after WPI live

What type of attack did we perform

Assume the IV and the key of Trivium each consist of 80 all zero bits Com pute the first 70 bits s1 s70 during the warm up phase of Trivium Note that these are only internal bits which are not used for encryption since the warm up phase lasts for clock cycles

The Data Encryption Standard DES and Alternatives

The Data Encryption Standard DES has been by far the most popular block ci pher for most of the last 30 years Even though it is nowadays not considered secure against a determined attacker because the DES key space is too small it is still used in legacy applications Furthermore encrypting data three times in a row with DES a process referred to as 3DES or triple DES yields a very secure cipher which is still widely used today Section deals with 3DES Perhaps what is more important since DES is by far the best studied symmetric algorithm its de sign principles have inspired many current ciphers Hence studying DES helps us to understand many other symmetric algorithms

In this chapter you will learn

The design process of DES which is very helpful for understanding the technical and political evolution of modern cryptography

Basic design ideas of block ciphers including confusion and diffusion which are important properties of all modern block ciphers

The internal structure of DES including Feistel networks S boxes and the key schedule

Security analysis of DES

Alternatives to DES including 3DES

C Paar J Pelzl Understanding Cryptography 55

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

In a mildly revolutionary act was performed by the US National Bureau of Standards NBS which is now called National Institute of Standards and Tech nology NIST the NBS initiated a request for proposals for a standardized cipher in the USA The idea was to find a single secure cryptographic algorithm which could be used for a variety of applications Up to this point in time governments had always considered cryptography and in particular cryptanalysis so crucial for na tional security that it had to be kept secret However by the early s the demand for encryption for commercial applications such as banking had become so pressing that it could not be ignored without economic consequences

The NBS received the most promising candidate in from a team of cryp tographers working at IBM The algorithm IBM submitted was based on the cipher Lucifer Lucifer was a family of ciphers developed by Horst Feistel in the late s and was one of the first instances of block ciphers operating on digital data Lucifer is a Feistel cipher which encrypts blocks of 64 bits using a key size of bits In order to investigate the security of the submitted ciphers the NBS requested the help of the National Security Agency NSA which did not even admit its existence at that point in time It seems certain that the NSA influenced changes to the cipher which was rechristened DES One of the changes that occurred was that DES is specifically designed to withstand differential cryptanalysis an attack not known to the public until It is not clear whether the IBM team developed the knowl edge about differential cryptanalysis by themselves or whether they were guided by the NSA Allegedly the NSA also convinced IBM to reduce the Lucifer key length of bit to 56 bit which made the cipher much more vulnerable to brute force attacks

The NSA involvement worried some people because it was feared that a secret trapdoor i e a mathematical property with which DES could be broken but which is only known to NSA might have been the real reason for the modifications Another major complaint was the reduction of the key size Some people conjectured that the NSA would be able to search through a key space of thus breaking it by brute force In later decades most of these concerns turned out to be unfounded Section provides more information about real and perceived security weaknesses of DES

Despite of all the criticism and concerns in the NBS finally released all specifications of the modified IBM cipher as the Data Encryption Standard FIPS PUB 46 to the public Even though the cipher is described down to the bit level in the standard the motivation for parts of the DES design the so called design crite ria especially the choice of the substitution boxes were never officially released

With the rapid increase in personal computers in the early s and all specifica tions of DES being publicly available it become easier to analyze the inner structure of the cipher During this period the civilian cryptography research community also grew and DES underwent major scrutiny However no serious weaknesses were found until Originally DES was only standardized for 10 years until Due to the wide use of DES and the lack of security weaknesses the NIST reaf

Introduction to DES 57

firmed the federal use of the cipher until when it was finally replaced by the

Advanced Encryption Standard AES

Confusion and Diffusion

Before we start with the details of DES it is instructive to look at primitive op erations which can be applied in order to achieve strong encryption According to the famous information theorist Claude Shannon there are two primitive operations with which strong encryption algorithms can be built

Confusion is an encryption operation where the relationship between key and ciphertext is obscured Today a common element for achieving confusion is sub stitution which is found in both DES and AES

Diffusion is an encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical proper ties of the plaintext A simple diffusion element is the bit permutation which is used frequently within DES AES uses the more advanced Mixcolumn operation

Ciphers which only perform confusion such as the Shift Cipher cf Sect 3 or the World War II encryption machine Enigma are not secure Neither are ci phers which only perform diffusion However through the concatenation of such operations a strong cipher can be built The idea of concatenating several encryp tion operation was also proposed by Shannon Such ciphers are known as product ciphers All of today s block ciphers are product ciphers as they consist of rounds which are applied repeatedly to the data Fig

Principle of an N round product cipher where each round performs a confusion and diffusion operation

Modern block ciphers possess excellent diffusion properties On a cipher level this means that changing of one bit of plaintext results on average in the change of

half the output bits i e the second ciphertext looks statistically independent of the first one This is an important property to keep in mind when dealing with block ciphers We demonstrate this behavior with the following simple example

Example Let s assume a small block cipher with a block length of 8 bits En cryption of two plaintexts x1 and x2 which differ only by one bit should roughly result in something as shown in Fig

Principle of diffusion of a block cipher

Note that modern block ciphers have block lengths of 64 or bit but they show exactly the same behavior if one input bit is flipped

Overview of the DES Algorithm

DES is a cipher which encrypts blocks of length of 64 bits with a key of size of 56 bits Fig 3

DES is a symmetric cipher i e the same same key is used for encryption and decryption DES is like virtually all modern block ciphers an iterative algorithm For each block of plaintext encryption is handled in 16 rounds which all perform the identical operation Figure shows the round structure of DES In every round a different subkey is used and all subkeys ki are derived from the main key k

Let s now have a more detailed view on the internals of DES as shown in Fig structure in the figure is called a Feistel network It can lead to very strong ciphers if carefully designed Feistel networks are used in many but cer tainly not in all modern block ciphers In fact AES is not a Feistel cipher In addition to its potential cryptographic strength one advantage of Feistel networks is that encryption and decryption are almost the same operation Decryption requires

Overview of the DES Algorithm 59

Iterative structure of DES

only a reversed key schedule which is an advantage in software and hardware im plementations We discuss the Feistel network in the following

After the initial bitwise permutation IP of a 64 bit plaintext x the plaintext is split into two halves L0 and R0 These two 32 bit halves are the input to the Feistel network which consists of 16 rounds The right half Ri is fed into the function

f The output of the f function is XORed as usually denoted by the symbol with the left 32 bit half Li Finally the right and left half are swapped This process repeats in the next round and can be expressed as

Ri Li 1 f Ri 1 ki

where i 1 16 After round 16 the 32 bit halves L16 and R16 are swapped again

and the final permutation IP 1 is the last operation of DES As the notation suggests the final permutation IP 1 is the inverse of the initial permutation IP In each round a round key ki is derived from the main 56 bit key using what is called the key

It is crucial to note that the Feistel structure really only encrypts decrypts half of the input bits per each round namely the left half of the input The right half is copied to the next round unchanged In particular the right half is not encrypted with the f function In order to get a better understanding of the working of Feistel

cipher the following interpretation is helpful Think of the f function as a pseu dorandom generator with the two input parameters Ri 1 and ki The output of the pseudorandom generator is then used to encrypt the left half Li 1 with an XOR op

eration As we saw in if the output of the f function is not predictable for an attacker this results in a strong encryption method

The Feistel structure of DES

The two aforementioned basic properties of ciphers i e confusion and diffusion are realized within the f function In order to thwart advanced analytical attacks the f function must be designed extremely carefully Once the f function has been designed securely the security of a Feistel cipher increases with the number of key bits used and the number of rounds

Before we discuss all components of DES in detail here is an algebraic descrip tion of the Feistel network for the mathematically inclined reader The Feistel struc ture of each round bijectively maps a block of 64 input bits to 64 output bits i e every possible input is mapped uniquely to exactly one output and vice versa This mapping remains bijective for some arbitrary function f i e even if the embedded function f is not bijective itself In the case of DES the function f is in fact a sur

jective many to one mapping It uses nonlinear building blocks and maps 32 input bits to 32 output bits using a 48 bit round key ki with 1 i 16

Internal Structure of DES

The structure of DES as depicted in Fig shows the internal functions which we will discuss in this section The building blocks are the initial and final permutation the actual DES rounds with its core the f function and the key schedule

Initial and Final Permutation

As shown in Figs and the initial permutation IP and the final permuta tion IP 1 are bitwise permutations A bitwise permutation can be viewed as simple crosswiring Interestingly permutations can be very easily implemented in hardware

but are not particularly fast in software Note that both permutations do not increase the security of DES at all The exact rationale for the existence of these two permu tations is not known but it seems likely that their original purpose was to arrange the plaintext ciphertext and bits in a bytewise manner to make data fetches easier for 8 bit data busses which were the state of the art register size in the early s

Examples for the bit swaps of the initial permutation

Examples for the bit swaps of the final permutation

The details of the transformation IP are given in Fig This table like all other tables in this chapter should be read from left to right top to bottom The table indicates that input bit 58 is mapped to output position 1 input bit 50 is mapped to

the second output position and so forth The final permutation IP 1 performs the inverse operation of IP as shown in Fig

Initial permutation IP

Final permutation IP 1

As mentioned earlier the f function plays a crucial role for the security of DES In round i it takes the right half Ri 1 of the output of the previous round and the current round key ki as input The output of the f function is used as an XOR mask for encrypting the left half input bits Li 1

0 Block diagram of the f function

The structure of the f function is shown in Fig First the 32 bit input is ex panded to 48 bits by partitioning the input into eight 4 bit blocks and by expanding each block to 6 bits This happens in the E box which is a special type of permuta tion The first block consists of the bits 1 2 3 4 the second one of 5 6 7 8 etc The expansion to six bits can be seen in Fig

1 Examples for the bit swaps of the expansion function E

As can be seen from the Table exactly 16 of the 32 input bits appear twice in the output However an input bit never appears twice in the same 6 bit output block The expansion box increases the diffusion behavior of DES since certain input bits influence two different output locations

Expansion permutation E

Next the 48 bit result of the expansion is XORed with the round key ki and the eight 6 bit blocks are fed into eight different substition boxes which are often referred to as S boxes Each S box is a lookup table that maps a 6 bit input to a 4 bit output Larger tables would have been cryptographically better but they also become much larger eight 4 by 6 tables were probably close the maximum size which could be fit on a single integrated circuit in Each S box contains 26 64 entries which are typically represented by a table with 16 columns and 4 rows Each entry is a 4 bit value All S boxes are listed in Tables to Note that all S boxes are different The tables are to be read as indicated in Fig the most significant bit MSB and the least significant bit LSB of each 6 bit input select the row of the table while the four inner bits select the column The integers 15 of each entry in the table represent the decimal notation of a 4 bit value

Example The S box input b indicates the row 3 i e fourth row numbering starts with and the column 2 2 i e the third column If the input b is fed into S box 1 the output is S 7 8 2

2 Example of the decoding of the input by S box 1

The S boxes are the core of DES in terms of cryptographic strength They are the only nonlinear element in the algorithm and provide confusion Even though the entire specification of DES was released by NBS NIST in the motivation for the choice of the S box tables was never completely revealed This often gave rise

to speculation in particular with respect to the possible existence of a secret back door or some other intentionally constructed weakness which could be exploited by the NSA However now we know that the S boxes were designed according to the criteria listed below

Each S box has six input bits and four output bits

No single output bit should be too close to a linear combination of the input bits

If the lowest and the highest bits of the input are fixed and the four middle bits are varied each of the possible 4 bit output values must occur exactly once

If two inputs to an S box differ in exactly one bit their outputs must differ in at least two bits

If two inputs to an S box differ in the two middle bits their outputs must differ in at least two bits

If two inputs to an S box differ in their first two bits and are identical in their last two bits the two outputs must be different

For any nonzero 6 bit difference between inputs no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference

A collision zero output difference at the 32 bit output of the eight S boxes is only possible for three adjacent S boxes

Note that some of these design criteria were not revealed until the s More information about the issue of the secrecy of the design criteria is found in Sect The S boxes are the most crucial elements of DES because they introduce a non

linearity to the cipher i e

S a S b S a b

Without a nonlinear building block an attacker could express the DES input and output with a system of linear equations where the key bits are the unknowns Such systems can easily be solved a fact that was used in the LFSR attack in Sect 2 However the S boxes were carefully designed to also thwart advanced mathematical attacks in particular differential cryptanalysis Interestingly differential cryptanal ysis was first discovered in the research community in At this point the IBM team declared that the attack was known to the designers at least 16 years earlier and that DES was especially designed to withstand differential cryptanalysis

Finally the 32 bit output is permuted bitwise according to the P permutation which is given in Table Unlike the initial permutation IP and its inverse IP 1 the permutation P introduces diffusion because the four output bits of each S box are permuted in such a way that they affect several different S boxes in the follow ing round The diffusion caused by the expansion S boxes and the permutation P

guarantees that every bit at the end of the fifth round is a function of every plaintext bit and every key bit This behavior is known as the avalanche effect

0 The permutation P within the f function

The key schedule derives 16 round keys ki each consisting of 48 bits from the original 56 bit key Another term for round key is subkey First note that the DES input key is often stated as 64 bit where every eighth bit is used as an odd parity bit over the preceding seven bits It is not quite clear why DES was specified that way In any case the eight parity bits are not actual key bits and do not increase the security DES is a 56 bit cipher not a 64 bit one

As shown in Fig the 64 bit key is first reduced to 56 bits by ignoring every eighth bit i e the parity bits are stripped in the initial PC 1 permutation Again the parity bits certainly do not increase the key space The name PC 1 stands for permuted choice one The exact bit connections that are realized by PC 1 are given in Table

3 Location of the eight parity bits for a 64 bit input key

1 Initial key permutation PC 1

The resulting 56 bit key is split into two halves C0 and D0 and the actual key schedule starts as shown in Fig The two 28 bit halves are cyclically shifted i e rotated left by one or two bit positions depending on the round i according to the following rules

In rounds i 1 2 9 16 the two halves are rotated left by one bit

In the other rounds where i 1 2 9 16 the two halves are rotated left by two bits

Note that the rotations only take place within either the left or the right half The total number of rotation positions is 28 This leads to the interesting property that C0 C16 and D0 D16 This is very useful for the decryption key

schedule where the subkeys have to be generated in reversed order as we will see in Sect

4 Key schedule for DES encryption

To derive the 48 bit round keys ki the two halves are permuted bitwise again with PC 2 which stands for permuted choice 2 PC 2 permutes the 56 input bits coming from Ci and Di and ignores 8 of them The exact bit connections of

PC 2 are given in Table

2 Round key permutation PC 2

Note that every round key is a selection of 48 permuted bits of the input key k The key schedule is merely a method of realizing the 16 permutations systemati cally Especially in hardware the key schedule is very easy to implement The key schedule is also designed so that each of the 56 key bits is used in different round keys each bit is used in approximately 14 of the 16 round keys

One advantage of DES is that decryption is essentially the same function as en cryption This is because DES is based on a Feistel network Figure shows a block diagram for DES decryption Compared to encryption only the key schedule is reversed i e in decryption round 1 subkey 16 is needed in round 2 subkey 15 etc Thus when in decryption mode the key schedule algorithm has to generate the round keys as the sequence k16 k15 k1

Reversed Key Schedule

The first question that we have to clarify is how given the initial DES key k can we easily generate k16 Note that we saw above that C0 C16 and D0 D16 Hence k16 can be directly derived after PC 1

k16 PC 2 C16 D16

To compute k15 we need the intermediate variables C15 and D15 which can be de rived from C16 D16 through cyclic right shifts RS

k15 PC 2 C15 D15

PC 2 RS2 C16 RS2 D16

PC 2 RS2 C0 RS2 D0

The subsequent round keys k14 k13 k1 are derived via right shifts in a similar fashion The number of bits shifted right for each round key in decryption mode

In decryption round 1 the key is not rotated

In decryption rounds 2 9 and 16 the two halves are rotated right by one bit

In the other rounds 3 4 5 6 7 8 10 11 12 13 14 and 15 the two halves are rotated right by two bits

Figure shows the reversed key schedule for decryption

Decryption in Feistel Networks

We have not addressed the core question Why is the decryption function essentially the same as the encryption function The basic idea is that the decryption function reverses the DES encryption in a round by round manner That means that decryp tion round 1 reverses encryption round 16 decryption round 2 reverses encryption round 15 and so on Let s first look at the initial stage of decryption by looking at Fig Note that the right and left halves are swapped in the last round of DES

Ld Rd IP Y IP IP 1 R16 L16 R16 L16

6 Reversed key schedule for decryption of DES

Note that all variables in the decryption routine are marked with the superscript d whereas the encryption variables do not have superscripts The derived equation simply says that the input of the first round of decryption is the output of the last round of encryption because final and initial permutations cancel each other out We will now show that the first decryption round reverses the last encryption round For this we have to express the output values Ld Rd of the first decryption round 1

in terms of the input values of the last encryption round L15 R15 The first one is easy

We now look at how Rd is computed

Rd Ld f Rd k16 R16 f L16 k16

Rd L15 f R15 k16 f R15 k16

Rd L15 f R15 k16 f R15 k16 L15

The crucial step is shown in the last equation above An identical output of the

f function is XORed twice to L15 These operations cancel each other out so that

Rd L15 Hence after the first decryption round we in fact have computed the same values we had before the last encryption round Thus the first decryption round reverses the last encryption round This is an iterative process which continues in the next 15 decryption rounds and that can be expressed as

where i 0 1 16 In particular after the last decryption round

Finally at the end of the decryption process we have to reverse the initial per mutation

IP 1 Rd Ld IP 1 L0 R0 IP 1 IP x x

where x is the plaintext that was the input to the DES encryption

As we discussed in Sect 2 ciphers can be attacked in several ways With respect to cryptographic attacks we distinguish between exhaustive key search or brute force attacks and analytical attacks The latter was demonstrated with the LFSR attack in Sect 2 where we could easily break a stream cipher by solving a system of linear equations Shortly after DES was proposed two major criticisms against the cryptographic strength of DES centered around two arguments

The key space is too small i e the algorithm is vulnerable against brute force attacks

The design criteria of the S boxes was kept secret and there might have existed an analytical attack that exploits mathematical properties of the S boxes but which is only known to the DES designers

We discuss both types of attacks below However we also state the main con clusion about DES security already here Despite very intensive cryptanalysis over the lifetime of DES current analytical attacks are not very efficient However DES can relatively easily be broken with an exhaustive key search attack and thus plain DES is not suited for most applications any more

Exhaustive Key Search

The first criticism is nowadays certainly justified The original cipher proposed by IBM had a key length of bits and it is suspicious that it was reduced to 56 bits The official statement that a cipher with a shorter key length made it easier to im plement the DES algorithm on a single chip in does not sound too convincing For clarification let s recall the principle of an exhaustive key search or brute force attack

Note that there is a small chance of 16 that an incorrect key is found i e a key k which decrypts only the one ciphertext y correctly but not subsequent ciphertexts If one wants to rule out this possibility an attacker must check such a key candidate with a second plaintext ciphertext pair More about this is found in Sect

Regular computers are not particularly well suited to perform the key tests necessary but special purpose key search machines are an option It seems highly likely that large government institutions have long been able to build such brute force crackers which can break DES in a matter of days In Whitfield Diffie and Martin Hellman 59 estimated that it was possible to build an exhaustive key search machine for approximately 00 Even though they later stated that their cost estimate had been too optimistic it was clear from the beginning that a cracker could be built with sufficient funding

At the rump session of the CRYPTO conference Michael Wiener proposed the design of a very efficient key search machine which used pipelining techniques An update of his proposal can be found in He estimated the cost of his de sign at approximately 00 and the time required to find the key at days This was a proposal only and the machine was not built In however the EFF Electronic Frontier Foundation built the hardware machine Deep Crack which performed a brute force attack against DES in 56 hours Figure shows a photo of Deep Crack The machine consisted of integrated circuits where each had 24 key test units The average search time of Deep Crack was 15 days and the ma chine was built for less than 0 The successful break with Deep Crack was considered the official demonstration that DES is no longer secure against deter mined attacks by many people Please note that this break does not imply that a weak algorithm had been in use for more than 20 years It was only possible to build Deep Crack at such a relatively low price because digital hardware had become

cheap In the s it would have been impossible to build a DES cracker with out spending many millions of dollars It can be speculated that only government agencies were willing to spend such an amount of money for code breaking

7 Deep Crack the hardware exhaustive key search machine that broke DES in reproduced with permission from Paul Kocher

DES brute force attacks also provide an excellent case study for the continuing decrease in hardware costs In the COPACOBANA Cost Optimized Parallel Code Breaker machine was built based on commercial integrated circuits by a team of researchers from the Universities of Bochum and Kiel in Germany the authors of this book were heavily involved in this effort COPACOBANA allows one to break DES with an average search time of less than 7 days The interesting part of this undertaking is that the machine could be built with hardware costs in the 00 range Figure shows a picture of COPACOBANA

8 COPACOBANA A cost optimized parallel code breaker

In summary a key size of 56 bits is too short to encrypt confidential data nowa days Hence single DES should only be used for applications where only short term security is needed say a few hours or where the value of the encrypted data is very low However variants of DES in particular 3DES are still secure

Implementation in Software and Hardware 75

As was shown in the first chapter analytical attacks can be very powerful Since the introduction of DES in the mid s many excellent researchers in academia and without doubt many excellent researchers in intelligence agencies tried to find weaknesses in the structure of DES which allowed them to break the cipher It is a major triumph for the designers of DES that no weakness was found until

cipher However it turned out that the DES S boxes are particularly resistant against this attack In fact one member of the original IBM design team declared after the discovery of DC that they had been aware of the attack at the time of design Al legedly the reason why the S box design criteria were not made public was that the design team did not want to make such a powerful attack public If this claim is true

and all circumstances support it it means that the IBM and NSA team was 15 years ahead of the research community It should be noted however that in the s and s relatively few people did active research in cryptography

In a related but distinct analytical attack was published by Mitsuru Matsui which was named linear cryptanalysis LC Similar to differential cryptanalysis the effectiveness of this attack also heavily depends on the structure of the S boxes

What is the practical relevance of these two analytical attacks against DES It turns out that an attacker needs plaintext ciphertext pairs for a successful DC attack This assumes particularly chosen plaintext blocks for random plaintext pairs are needed In the case of LC an attacker needs plaintext ciphertext pairs All these numbers seem highly impractical for several reasons First an attacker needs to know an extremely large number of plaintexts i e pieces of data which are supposedly encrypted and thus hidden from the attacker Second collecting and storing such an amount of data takes a long time and requires considerable memory resources Third the attack only recovers one key This is actually one of many arguments for introducing key freshness in cryptographic applications As a result of all these arguments it does not seem likely that DES can be broken with either DC or LC in real world systems However both DC and LC are very powerful attacks which are applicable to many other block ciphers Table provides an overview of proposed and realized attacks against DES over the last three decades Some entries refer to what is known as the DES Challenges Starting in several DES breaking challenges were organized by the company RSA Security

Implementation in Software and Hardware

In the following we provide a brief assessment of DES implementation properties in software and hardware When we talk about software we refer to DES implemen tations running on desktop CPUs or embedded microprocessors like smart cards

3 History of full round DES attacks

or cell phones Hardware refers to DES implementations running on ICs such as application specific integrated circuits ASICs or field programmable gate arrays FPGAs

A straightforward software implementation which follows the data flow of most DES descriptions such as the one presented in this chapter results in a very poor performance This is due to the fact that many of the atomic DES operations involve bit permutation in particular the E and P permutation which are slow in software Similarly small S boxes such as used in DES are efficient in hardware but only mod erately efficient on modern CPUs There have been numerous methods proposed for accelerating DES software implementations The general idea is to use tables with precomputed values of several DES operations e g of several S boxes and the per mutation Optimized implementations require about cycles for encrypting one block on a 32 bit CPU On a 2 GHz CPU this translates into a theoretical throughput of about Mbits s 3DES which is considerably more secure than single DES runs at almost exactly of the DES speed Note that nonoptimized implementa tions are considerably slower often below Mbit s

A notable method for accelerating software implementations of DES is bit slicing developed by Eli Biham 20 On a MHz DEC Alpha workstation an encryption rate of Mbit sec has been reported which was much faster than a standard DES implementation at that time The limitation of bit slicing however is that several blocks are encrypted in parallel which can be a drawback for certain

modes of operation such as Cipher Block Chaining CBC and Output Feedback OFB mode cf

One design criterion for DES was its efficiency in hardware Permutations such as the E P IP and IP 1 permutations are very easy to implement in hardware as they only require wiring but no logic The small 6 by 4 S boxes are also relatively

easily realizable in hardware Typically they are implemented with Boolean logic i e logic gates On average one S box requires about gates

An area efficient implementation of a single DES round can be done with less than gates If a high throughput is desired DES can be implemented extremely fast by fitting multiple rounds in one circuit e g by using pipelining On modern ASICs and FPGAs throughput rates of several Gbit sec are possible On the other end of the performance spectrum very small implementations with fewer than gates even fit onto lowcost radio frequency identification RFID chips

There exist a wealth of other block ciphers Even though there are many ciphers which have security weaknesses or which are not well investigated there are also many block ciphers which appear very strong In the following a brief list of ciphers is given which can be of interest depending on the application needs

The Advanced Encryption Standard AES and the AES Finalist Ciphers

By now the algorithm of choice for many many applications has become the Ad vanced Encryption Standard AES which will be introduced in detail in the follow ing chapter AES is with its three key lengths of and bit secure against brute force attacks for several decades and there are no analytical attacks with any reasonable chance of success known

AES was the result of an open competition and in the last stage of the selection process there were four other finalist algorithms These are the block ciphers Mars RC6 Serpent and Twofish All of them are cryptographically strong and quite fast especially in software Based on today s knowledge they can all be recommended Mars Serpent and Twofish can be used royalty free

Triple DES 3DES and DESX

An alternative to AES or the AES finalist algorithms is triple DES often denoted as

3DES 3DES consists of three subsequent DES encryptions

y DESk3 DESk2 DESk1 x

with different keys as shown in Fig

3DES seems resistant to both brute force attacks and any analytical attack imag inable at the moment See for more information on double and triple en cryption Another version of 3DES is

y DESk DES 1 DESk x

The advantage here is that 3DES performs single DES encryption if k3 k2 k1 which is sometimes desired in implementations that should also support single DES for legacy reasons 3DES is very efficient in hardware but not particularly in soft ware It is popular in financial applications as well as for protecting biometric infor mation in electronic passports

A different approach for strengthening DES is to use key whitening For this two additional 64 bit keys k1 and k2 are XORed to the plaintext and ciphertext respec tively prior to and after the DES algorithm This yields the following encryption scheme

y DESk k1 k2 x DESk x k1 k2

This surprisingly simple modification makes DES much more resistant against ex haustive key searches More about key whitening is said in Sect 3

Lightweight Cipher PRESENT

Over the last few years several new block algorithms which are classified as lightweight ciphers have been proposed Lightweight commonly refers to algo rithms with a very low implementation complexity especially in hardware Trivium Sect 3 is an example of a lightweight stream cipher A promising block cipher candidate is PRESENT which was designed specifically for applications such as

RFID tags or other pervasive computing applications that are extremely power or cost constrained One of the book authors participated in the design of PRESENT

for i l to 3l do addRoundKey STATE Ki sBoxLayer STATE pLayer STATE

addRoundKey STATE K32

0 Internal structure and pseudocode of the block cipher PRESENT

Unlike DES PRESENT is not based on a Feistel network Instead it is a substitution permutation network SP network and consists of 31 rounds The block length is 64 bits and two key lengths of 80 and bits are supported Each of the 31 rounds consists of an XOR operation to introduce a round key Ki for 1 i 32 where K32 is used after round 31 a nonlinear substitution layer sBoxLayer and a linear bitwise permutation pLayer The nonlinear layer uses a single 4 bit S box S which is applied 16 times in parallel in each round The key schedule generates 32 round keys from the user supplied key The encryption rou tine of the cipher is described in pseudocode in Fig and each stage is now specified in turn

addRoundKey At the beginning of each round the round key Ki is XORed to the current STATE

sBoxlayer PRESENT uses a single 4 bit to 4 bit S box This is a direct conse quence of the pursuit of hardware efficiency since such an S Box allows a much more compact implementation than e g an 8 bit S box The S box entries in hex adecimal notation are given in Table

4 The PRESENT S box in hexadecimal notation

The 64 bit data path b63 b0 is referred to as state For the sBoxLayer the cur

rent state is considered as sixteen 4 bit words w15 w0 where wi b4 i 3 b4 i 2 b4 i 1 b4 i for 0 i 15 and the output are the 16 words S wi

pLayer Just like DES the mixing layer was chosen as a bit permutation which can be implemented extremely compactly in hardware The bit permutation used in PRESENT is given by Table Bit i of STATE is moved to bit position P i

5 The permutation layer of PRESENT

The bit permutation is quite regular and can in fact be expressed in the following way

P i i 16 mod 63 i 0 62

Key Schedule We describe in the following the key schedule for PRESENT with an 80 bit key Since the main applications of PRESENT are low cost systems this key length is in most cases appropriate Details of the key schedule for PRESENT can be found in 29 The user supplied key is stored in a key register K and is represented as k 8 k0 At round i the 64 bit round key Ki 2 0 consists of the 64 leftmost bits of the current contents of register K Thus at round i we have

Ki 2 0 k 8 k16

The first subkey K1 is a direct copy of 64 bit of the user supplied key For the fol lowing subkeys K2 K32 the key register K k 8 k0 is updated as follows

k 8 k k 7 k 9

k 8k 5 k 8k 5 round counter

Thus the key schedule consists of three operations 1 the key register is ro tated by 61 bit positions to the left 2 the leftmost four bits are passed through the PRESENT S box and 3 the round counter value i is XORed with bits k 8k 5 of K where the least significant bit of round counter is on the right This counter is a simple integer which takes the values 1 0

1 Note that for the derivation of K2 the counter value 1 is used for K3

the counter value 0 and so on

Implementation As a result of the aggressively hardware optimized design of PRESENT its software performance is not very competitive relative to modern ci phers like AES An optimized software implementation on a Pentium III CPU in

Discussion and Further Reading 81

C achieves a throughput of about 60 Mbit s at a frequency of 1 GHz However it performs quite well on small microprocessors which are common in inexpensive consumer products

PRESENT 80 can be implemented in hardware with area requirements of ap proximately gate equivalences where the encryption of one 64 bit plain text block requires 32 clock cycles As an example at a clock rate of 1 MHz which is quite typical on low cost devices a throughput of 2 Mbit s is achieved which is sufficient for most such applications It is possible to realize the cipher with as few as approximately gate equivalences where the encryption of one 64 bit plain text requires clock cycles A fully pipelined implementation of PRESENT with 31 encryption stages achieves a throughput of 64 bit per clock cycle which can be tranlsated into encryption throughputs of more than 50 Gbit s

Even though no attacks against PRESENT are known at the time of writing it should be noted that it is a relatively new block cipher

Discussion and Further Reading

DES History and Attacks Even though plain DES i e non 3DES is today mainly used in legacy applications its history helps us understand the evolution of cryptography since the mid s from an obscure discipline almost solely stud ied in government organizations towards an open discipline with many players in industry and academia A summary of the DES history can be found in The two main analytical attacks developed against DES differential and linear crypt analysis are today among the most powerful general methods for breaking block ciphers Readers interested in the theory of block ciphers are encouraged to study these attacks Good descriptions are given in 21

As we have seen in this chapter DES should no longer be used since a brute force attack can be accomplished at low cost in little time with cryptanalytical hardware The two machines built outside governments Deep Crack and COPACOBANA are instructive examples of how to build low cost supercomputers for very narrowly defined computational tasks More information about Deep Crack can be found on the Internet 78 and about COPACOBANA in the articles 88 and online at 47 Readers interested in the fascinating area of cryptanalytical computers in general should take a look at the SHARCS Special purpose Hardware for Attacking Cryptographic Systems workshop series which started in and has information online

DES Alternatives It should be noted that hundreds of block ciphers have been proposed over the last three decades especially in the late s and in the s DES has influenced the design of many other encryption algorithms It is probably fair to say that the majority of today s successful block ciphers have borrowed ideas from DES Some of the popular block ciphers are also based on Feistel networks as is DES Examples of Feistel ciphers include Blowfish CAST KASUMI Mars

MISTY1 Twofish and RC6 One cipher which is well known and markedly different from DES is IDEA it uses arithmetic in three different algebraic structures as atomic operations

DES is a good example of a block cipher which is very efficient in hardware The recent advent of pervasive computing has created a need for extremely small ciphers for applications such as RFID tags or low cost smart cards e g for high volume public transportation payment tickets Good references for PRESENT are 29 In addition to PRESENT other recently proposed very small block ciphers include Clefia 48 HIGHT 93 and mCrypton A good overview of the new field of lightweight cryptography is given in the surveys 71 98 A more in depth treatment of lightweight algorithms can be found in the Ph D dissertation

Implementation With respect to software implementation of DES an early refer ence is 20 More advanced techniques are described in The powerful method of bit slicing is applicable not only to DES but to most other ciphers

Regarding DES hardware implementation an early but still very interesting ref erence is There are many descriptions of high performance implementations of DES on a variety of hardware platforms including FPGAs standard ASICs as well as more exotic semiconductor technology 67

DES was the dominant symmetric encryption algorithm from the mid s to the mid s Since 56 bit keys are no longer secure the Advanced Encryption Standard AES was created

Standard DES with 56 bit key length can be broken relatively easily nowadays through an exhaustive key search

DES is quite robust against known analytical attacks In practice it is very diffi cult to break the cipher with differential or linear cryptanalysis

DES is reasonably efficient in software and very fast and small in hardware

By encrypting with DES three times in a row triple DES 3DES is created against which no practical attack is currently known

The default symmetric cipher is nowadays often AES In addition the other four AES finalist ciphers all seem very secure and efficient

Since about several proposals for lightweight ciphers have been made They are suited for resource constrained applications

As stated in Sect one important property which makes DES secure is that the S boxes are nonlinear In this problem we verify this property by computing the output of S1 for several pairs of inputs

Show that S1 x1 S1 x2 S1 x1 x2 where denotes bitwise XOR for

We want to verify that IP and IP 1 are truly inverse operations We con sider a vector x x1 x2 x64 of 64 bit Show that IP 1 IP x x for the first five bits of x i e for xi i 1 2 3 4 5

What is the output of the first round of the DES algorithm when the plaintext and the key are both all zeros

What is the output of the first round of the DES algorithm when the plaintext and the key are both all ones

Remember that it is desirable for good block ciphers that a change in one input bit affects many output bits a property that is called diffusion or the avalanche effect We try now to get a feeling for the avalanche property of DES We apply an input word that has a 1 at bit position 57 and all other bits as well as the key are zero Note that the input word has to run through the initial permutation

How many S boxes get different inputs compared to the case when an all zero plaintext is provided

What is the minimum number of output bits of the S boxes that will change according to the S box design criteria

What is the output after the first round

How many output bit after the first round have actually changed compared to the case when the plaintext is all zero Observe that we only consider a single round here There will be more and more output differences after every new round Hence the term avalanche effect

An avalanche effect is also desirable for the key A one bit change in a key should result in a dramatically different ciphertext if the plaintext is unchanged

Assume an encryption with a given key Now assume the key bit at position 1 prior to PC 1 is being flipped Which S boxes in which rounds are affected by the bit flip during DES encryption

Which S boxes in which DES rounds are affected by this bit flip during DES decryption

A DES key Kw is called a weak key if encryption and decryption are identical operations

DESKw x DESK 1 x for all x

Describe the relationship of the subkeys in the encryption and decryption algo rithm that is required so that Eq is fulfilled

There are four weak DES keys What are they

What is the likelihood that a randomly selected key is weak

DES has a somewhat surprising property related to bitwise complements of its inputs and outputs We investigate the property in this problem

We denote the bitwise complement of a number A that is all bits are flipped by

Al Let denote bitwise XOR We want to show that if

This states that if we complement the plaintext and the key then the ciphertext output will also be the complement of the original ciphertext Your task is to prove this property

Try to prove this property using the following steps

Show that for any bit strings A B of equal length

These two operations are needed for some of the following steps

Show that PC 1 kl PC 1 k l

Show that LSi Cil 1 LSi Ci 1 l

Using the two res ults from above show that if ki are the keys generated from k

then kil are the keys generated from kl where i 1 2 16

Show that IP xl IP x l

Show that E Rli E Ri l

Using all previous results show that if Ri 1 Li 1 ki generate Ri then Rli 1 Lil 1 kil

Show that Eq is true

Assume we perform a known plaintext attack against DES with one pair of plaintext and ciphertext How many keys do we have to test in a worst case sce nario if we apply an exhaustive key search in a straightforward way How many on average

In this problem we want to study the clock frequency requirements for a hard ware implementation of DES in real world applications The speed of a DES im plementation is mainly determined by the time required to do one core iteration This hardware kernel is then used 16 consecutive times in order to generate the en crypted output An alternative approach would be to build a hardware pipeline with 16 stages resulting in 16 fold increased hardware costs

Let s assume that one core iteration can be performed in one clock cycle De velop an expression for the required clock frequency for encrypting a stream of data with a data rate r bit sec Ignore the time needed for the initial and final permutation

What clock frequency is required for encrypting a fast network link running at a speed of 1 Gb sec What is the clock frequency if we want to support a speed of 8 Gb sec

As the example of COPACOBANA shows key search machines need not be prohibitive from a monetary point of view We now consider a simple brute force attack on DES which runs on COPACOBANA

Compute the runtime of an average exhaustive key search on DES assuming the following implementational details

COPACOBANA platform with 20 FPGA modules

6 FPGAs per FPGA module

4 DES engines per FPGA

Each DES engine is fully pipelined and is capable of performing one encryp tion per clock cycle

MHz clock frequency

How many COPACOBANA machines do we need in the case of an average search time of one hour

Why does any design of a key search machine constitute only an upper security threshold By upper security threshold we mean a complexity measure which describes the maximum security that is provided by a given cryptographic algo rithm

We study a real world case in this problem A commercial file encryption program from the early s used standard DES with 56 key bits In those days performing an exhaustive key search was considerably harder than nowadays and thus the key length was sufficient for some applications Unfortunately the imple mentation of the key generation was flawed which we are going to analyze Assume that we can test keys per second on a conventional PC

The key is generated from a password consisting of 8 characters The key is a simple concatenation of the 8 ASCII characters yielding 64 key bits With the permutation PC 1 in the key schedule the least significant bit LSB of each 8 bit character is ignored yielding 56 key bits

What is the size of the key space if all 8 characters are randomly chosen 8 bit ASCII characters How long does an average key search take with a single PC

How many key bits are used if the 8 characters are randomly chosen 7 bit ASCII characters i e the most significant bit is always zero How long does an aver age key search take with a single PC

How large is the key space if in addition to the restriction in Part 2 only let ters are used as characters Furthermore unfortunately all letters are converted

to capital letters before generating the key in the software How long does an average key search take with a single PC

This problem deals with the lightweight cipher PRESENT

Calculate the state of PRESENT 80 after the execution of one round You can use the following table to solve this problem with paper and pencil Use the following values in hexadecimal notation

plaintext key BBBB EEEE FFFF

Now calculate the round key for the second round using the following table

The Advanced Encryption Standard AES

The Advanced Encryption Standard AES is the most widely used symmetric cipher today Even though the term Standard in its name only refers to US government applications the AES block cipher is also mandatory in several industry standards and is used in many commercial systems Among the commercial standards that include AES are the Internet security standard IPsec TLS the Wi Fi encryption standard IEEE i the secure shell network protocol SSH Secure Shell the Internet phone Skype and numerous security products around the world To date there are no attacks better than brute force known against AES

In this chapter you will learn

The design process of the US symmetric encryption standard AES

The encryption and decryption function of AES

The internal structure of AES namely

byte substitution layer

Basic facts about Galois fields

Efficiency of AES implementations

C Paar J Pelzl Understanding Cryptography 87

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

In the US National Institute of Standards and Technology NIST indicated that DES should only be used for legacy systems and instead triple DES 3DES should be used Even though 3DES resists brute force attacks with today s technol ogy there are several problems with it First it is not very efficient with regard to software implementations DES is already not particularly well suited for software and 3DES is three times slower than DES Another disadvantage is the relatively short block size of 64 bits which is a drawback in certain applications e g if one wants to built a hash function from a block cipher cf Sect 2 Finally if one is worried about attacks with quantum computers which might become reality in a few decades key lengths on the order of bits are desirable All these consider ation led NIST to the conclusion that an entirely new block cipher was needed as a replacement for DES

In NIST called for proposals for a new Advanced Encryption Standard AES Unlike the DES development the selection of the algorithm for AES was an open process administered by NIST In three subsequent AES evaluation rounds NIST and the international scientific community discussed the advantages and dis advantages of the submitted ciphers and narrowed down the number of potential candidates In NIST declared the block cipher Rijndael as the new AES and published it as a final standard FIPS PUB Rijndael was designed by two young Belgian cryptographers

Within the call for proposals the following requirements for all AES candidate submissions were mandatory

block cipher with bit block size

three key lengths must be supported and bit

security relative to other submitted algorithms

efficiency in software and hardware

The invitation for submitting suitable algorithms and the subsequent evaluation of the successor of DES was a public process A compact chronology of the AES selection process is given here

The need for a new block cipher was announced on January 2 by NIST

A formal call for AES was announced on September 12

Fifteen candidate algorithms were submitted by researchers from several coun tries by August 20

On August 9 five finalist algorithms were announced

Mars by IBM Corporation

RC6 by RSA Laboratories

Rijndael by Joan Daemen and Vincent Rijmen

Serpent by Ross Anderson Eli Biham and Lars Knudsen

Twofish by Bruce Schneier John Kelsey Doug Whiting David Wagner Chris Hall and Niels Ferguson

Overview of the AES Algorithm 89

On October 2 NIST announced that it had chosen Rijndael as the AES

On November 26 AES was formally approved as a US federal standard

It is expected that AES will be the dominant symmetric key algorithm for many commercial applications for the next few decades It is also remarkable that in the US National Security Agency NSA announced that it allows AES to encrypt classified documents up to the level SECRET for all key lengths and up to the TOP SECRET level for key lengths of either or bits Prior to that date only non public algorithms had been used for the encryption of classified documents

Overview of the AES Algorithm

The AES cipher is almost identical to the block cipher Rijndael The Rijndael block and key size vary between and bits However the AES standard only calls for a block size of bits Hence only Rijndael with a block length of bits is known as the AES algorithm In the remainder of this chapter we only discuss the standard version of Rijndael with a block length of bits

AES input output parameters

As mentioned previously three key lengths must be supported by Rijndael as this was an NIST design requirement The number of internal rounds of the cipher is a function of the key length according to Table

Key lengths and number of rounds for AES

key lengths rounds nr

In contrast to DES AES does not have a Feistel structure Feistel networks do not encrypt an entire block per iteration e g in DES 32 bits are encrypted

in one round AES on the other hand encrypts all bits in one iteration This is one reason why it has a comparably small number of rounds

AES consists of so called layers Each layer manipulates all bits of the data path The data path is also referred to as the state of the algorithm There are only three different types of layers Each round with the exception of the first consists of all three layers as shown in Fig the plaintext is denoted as x the ciphertext as y and the number of rounds as nr Moreover the last round nr does not make use of the MixColumn transformation which makes the encryption and decryption scheme symmetric

We continue with a brief description of the layers

Key Addition layer A bit round key or subkey which has been derived from the main key in the key schedule is XORed to the state

Byte Substitution layer S Box Each element of the state is nonlinearly trans formed using lookup tables with special mathematical properties This introduces confusion to the data i e it assures that changes in individual state bits propagate quickly across the data path

Diffusion layer It provides diffusion over all state bits It consists of two sublayers both of which perform linear operations

The ShiftRows layer permutes the data on a byte level

The MixColumn layer is a matrix operation which combines mixes blocks of four bytes

Similar to DES the key schedule computes round keys or subkeys k0 k1 knr

from the original AES key

Before we describe the internal functions of the layers in Sect we have to introduce a new mathematical concept namely Galois fields Galois field computa tions are needed for all operations within the AES layers

Some Mathematics A Brief Introduction to Galois Fields

In AES Galois field arithmetic is used in most layers especially in the S Box and the MixColumn layer Hence for a deeper understanding of the internals of AES we provide an introduction to Galois fields as needed for this purpose before we con tinue with the algorithm in Sect A background on Galois fields is not required for a basic understanding of AES and the reader can skip this section

Existence of Finite Fields

A finite field sometimes also called Galois field is a set with a finite number of elements Roughly speaking a Galois field is a finite set of elements in which we

round 1 Diffusion Layer

AES encryption block diagram

can add subtract multiply and invert Before we introduce the definition of a field we first need the concept of a a simpler algebraic structure a group

Roughly speaking a group is set with one operation and the corresponding in verse operation If the operation is called addition the inverse operation is subtrac tion if the operation is multiplication the inverse operation is division or multipli cation with the inverse element

Example The set of integers Zm 0 1 m 1 and the operation addition modulo m form a group with the neutral element 0 Every element a has an inverse a such that a a 0 mod m Note that this set does not form a group with the operation multiplication because most elements a do not have an inverse such that

In order to have all four basic arithmetic operations i e addition subtraction multiplication division in one structure we need a set which contains an additive and a multiplicative group This is what we call a field

Example The set R of real numbers is a field with the neutral element 0 for the additive group and the neutral element 1 for the multiplicative group Every real number a has an additive inverse namely a and every nonzero element a has a multiplicative inverse 1 a

In cryptography we are almost always interested in fields with a finite number of elements which we call finite fields or Galois fields The number of elements in the field is called the order or cardinality of the field Of fundamental importance is the following theorem

This theorem implies that there are for instance finite fields with 11 elements or with 81 elements since 81 34 or with elements since 28 and 2 is a prime However there is no finite field with 12 elements since 12 and 12 is thus not a prime power In the remainder of this section we look at how finite fields can be built and more importantly for our purpose how we can do arithmetic in them

The most intuitive examples of finite fields are fields of prime order i e fields with n 1 Elements of the field GF p can be represented by integers 0 1 p 1 The two operations of the field are modular integer addition and integer multiplication modulo p

This means that if we consider the integer ring Zm which was introduced in Sect 2 i e integers with modular addition and multiplication and m happens to be a prime Zm is not only a ring but also a finite field

In order to do arithmetic in a prime field we have to follow the rules for integer rings Addition and multiplication are done modulo p the additive inverse of any element a is given by a a 0 mod p and the multiplicative inverse of any

nonzero element a is defined as a a 1 1 Let s have a look at an example of a prime field

Example We consider the finite field GF 5 0 1 2 3 4 The tables below describe how to add and multiply any two elements as well as the additive and

multiplicative inverse

does not exist 1

multiplicative inverse of the field elements Using these tables we can perform all calculations in this field without using modular reduction explicitly

A very important prime field is GF 2 which is the smallest finite field that exists Let s have a look at the multiplication and addition tables for the field

Example Let s consider the small finite field GF 2 0 1 Arithmetic is sim ply done modulo 2 yielding the following arithmetic tables

addition multiplication

As we saw in on stream ciphers GF 2 addition i e modulo 2 addition is equivalent to an XOR gate What we learn from the example above is that GF 2 multiplication is equivalent to the logical AND gate The field GF 2 is important for AES

Extension Fields GF 2m

In AES the finite field contains elements and is denoted as GF 28 This field was chosen because each of the field elements can be represented by one byte For the S Box and MixColumn transforms AES treats every byte of the internal data

path as an element of the field GF 28 and manipulates the data by performing arithmetic in this finite field

However if the order of a finite field is not prime and 28 is clearly not a prime the addition and multiplication operation cannot be represented by addition and mul tiplication of integers modulo 28 Such fields with m 1 are called extension fields In order to deal with extension fields we need 1 a different notation for field ele ments and 2 different rules for performing arithmetic with the elements We will see in the following that elements of extension fields can be represented as poly nomials and that computation in the extension field is achieved by performing a certain type of polynomial arithmetic

In extension fields GF 2m elements are not represented as integers but as poly nomials with coefficients in GF 2 The polynomials have a maximum degree of m 1 so that there are m coefficients in total for every element In the field GF 28

which is used in AES each element A GF 28 is thus represented as

A x a a1x a0 ai GF 2 0 1

Note that there are exactly 28 such polynomials The set of these polyno mials is the finite field GF 28 It is also important to observe that every polynomial can simply be stored in digital form as an 8 bit vector

A a7 a6 a5 a4 a3 a2 a1 a0

In particular we do not have to store the factors x7 x6 etc It is clear from the bit positions to which power xi each coefficient belongs

Addition and Subtraction in GF 2m

Let s now look at addition and subtraction in extension fields The key addition layer of AES uses addition It turns out that these operations are straightforward They are simply achieved by performing standard polynomial addition and subtraction We merely add or subtract coefficients with equal powers of x The coefficient additions or subtractions are done in the underlying field GF 2

Note that we perform modulo 2 addition or subtraction with the coefficients As we saw in addition and subtraction modulo 2 are the same operation More over addition modulo 2 is equal to bitwise XOR Let s have a look at an example in the field GF 28 which is used in AES

Example Here is how the sum C x A x B x of two elements from GF 28

Note that if we computed the difference of the two polynomials A x B x from the example above we would get the same result as for the sum

Multiplication in GF 2m

Multiplication in GF 28 is the core operation of the MixColumn transformation of AES In a first step two elements represented by their polynomials of a finite field GF 2m are multiplied using the standard polynomial multiplication rule

A x B x am 1xm 1 a0 bm 1xm 1 b0 Ct x ct2m m 2 ct0

ct2m 2 am 1bm 1 mod 2

Note that all coefficients ai bi and ci are elements of GF 2 and that coeffi cient arithmetic is performed in GF 2 In general the product polynomial C x will have a degree higher than m 1 and has to be reduced The basic idea is an ap proach similar to the case of multiplication in prime fields in GF p we multiply the two integers divide the result by a prime and consider only the remainder Here is what we are doing in extension fields The product of the multiplication is divided by a certain polynomial and we consider only the remainder after the polynomial division We need irreducible polynomials for the module reduction We recall from Sect 1 that irreducible polynomials are roughly comparable to prime numbers i e their only factors are 1 and the polynomial itself

Thus every field GF 2m requires an irreducible polynomial P x of degree m with coefficients from GF 2 Note that not all polynomials are irreducible For example the polynomial x4 x3 x 1 is reducible since

x4 x3 x 1 x2 x 1 x2 1

and hence cannot be used to construct the extension field GF 24 Since primitive polynomials are a special type of irreducible polynomial the polynomials in Ta ble can be used for constructing fields GF 2m For AES the irreducible poly nomial

P x x8 x4 x3 x 1

is used It is part of the AES specification

Example We want to multiply the two polynomials A x x3 x2 1 and B x x2 x in the field GF 24 The irreducible polynomial of this Galois field is given as

The plain polynomial product is computed as

Ct x A x B x x5 x3 x2 x

We can now reduce Ct x using the polynomial division method we learned in school However sometimes it is easier to reduce each of the leading terms x4 and

x4 1 P x x 1

x4 x 1 mod P x x5 x2 x mod P x

Now we only have to insert the reduced expression for x5 into the intermediate result Ct x

C x x5 x3 x2 x mod P x C x x2 x x3 x2 x x3

It is important not to confuse multiplication in GF 2m with integer multiplica

tion especially if we are concerned with software implementations of Galois fields Recall that the polynomials i e the field elements are normally stored as bit vec tors in the computers If we look at the multiplication from the previous example the following very atypical operation is being performed on the bit level

x2 1 x2 x x3

This computation is not identical to integer arithmetic If the polynomials are in terpreted as integers i e 2 and 2 the result would have been 1 which is clearly not the same as the Galois field multipli cation product Hence even though we can represent field elements as integers data types we cannot make use of the integer arithmetic provided

Inversion in GF 28 is the core operation of the Byte Substitution transformation which contains the AES S Boxes For a given finite field GF 2m and the corre sponding irreducible reduction polynomial P x the inverse A 1 of a nonzero ele

ment A GF 2m is defined as

A 1 x A x 1 mod P x

For small fields in practice this often means fields with or fewer elements

lookup tables which contain the precomputed inverses of all field elements are often used Table shows the values which are used within the S Box of AES The table contains all inverses in GF 28 modulo P x x8 x4 x3 x 1 in hexadecimal notation A special case is the entry for the field element 0 for which

an inverse does not exist However for the AES S Box a substitution table is needed that is defined for every possible input value Hence the designers defined the S Box such that the input value 0 is mapped to the output value 0

Multiplicative inverse table in GF 28 for bytes xy used within the AES S Box

Example From Table the inverse of

x7 x6 x 2 C2 hex xy

is given by the element in row C column 2

2F hex 2 x5 x3 x2 x 1

This can be verified by multiplication

x7 x6 x x5 x3 x2 x 1 1 mod P x

Note that the table above does not contain the S Box itself which is a bit more

complex and will be described in Sect

As an alternative to using lookup tables one can also explicitly compute inverses The main algorithm for computing multiplicative inverses is the extended Euclidean algorithm which is introduced in Sect 1

Internal Structure of AES

In the following we examine the internal structure of AES Figure shows the graph of a single AES round The 16 byte input A0 A15 is fed byte wise into the

S Box The 16 byte output B0 B15 is permuted byte wise in the ShiftRows layer and mixed by the MixColumn transformation c x Finally the bit subkey ki is XORed with the intermediate result We note that AES is a byte oriented cipher

AES round function for rounds 1 2 nr 1

This is in contrast to DES which makes heavy use of bit permutation and can thus be considered to have a bit oriented structure

In order to understand how the data moves through AES we first imagine that the state A i e the bit data path consisting of 16 bytes A0 A1 A15 is arranged in a four by four byte matrix

As we will see in the following AES operates on elements columns or rows of the current state matrix Similarly the key bytes are arranged into a matrix with four rows and four bit key six bit key or eight bit key columns Here is as an example the state matrix of a bit key

We discuss now what happens in each of the layers

Byte Substitution Layer

As shown in Fig the first layer in each round is the Byte Substitution layer The Byte Substitution layer can be viewed as a row of 16 parallel S Boxes each with 8 input and output bits Note that all 16 S Boxes are identical unlike DES where eight different S Boxes are used In the layer each state byte Ai is replaced i e substituted by another byte Bi

The S Box is the only nonlinear element of AES i e it holds that ByteSub A ByteSub B ByteSub A B for two states A and B The S Box substitution is a bijective mapping i e each of the 28 possible input elements is one to one mapped to one output element This allows us to uniquely reverse the S Box which is needed for decryption In software implementations the S Box is usually realized as a by 8 bit lookup table with fixed entries as given in Table

AES S Box Substitution values in hexadecimal notation for input byte xy

Example Let s assume the input byte to the S Box is Ai C2 hex then the substituted value is

S C2 hex 25 hex

On a bit level and remember the only thing that is ultimate of interest in encryp tion is the manipulation of bits this substitution can be described as

Even though the S Box is bijective it does not have any fixed points i e there aren t any input values Ai such that S Ai Ai Even the zero input is not a fixed point S

Example Let s assume the input to the Byte Substitution layer is

in hexadecimal notation The output state is then

Mathematical description of the S Box For readers who are interested in how the S Box entries are constructed a more detailed description now follows This description however is not necessary for a basic understanding of AES and the remainder of this subsection can be skipped without problem Unlike the DES S Boxes which are essentially random tables that fulfill certain properties the AES S Boxes have a strong algebraic structure An AES S Box can be viewed as a two step mathematical transformation Fig

The two operations within the AES S Box which computes the function Bi S Ai

The first part of the substitution is a Galois field inversion the mathematics of which were introduced in Sect For each input element Ai the inverse is com puted Bti Ai 1 where both Ai and Bti are considered elements in the field GF 28

with the fixed irreducible polynomial P x x8 x4 x3 x 1 A lookup table

with all inverses is shown in Table Note that the inverse of the zero element does not exist However for AES it is defined that the zero element Ai 0 is mapped to itself

In the second part of the substitution each byte Bti is multiplied by a constant bit matrix followed by the addition of a constant 8 bit vector The operation is described

Note that Bt bt7 bt0 is the bitwise vector representation of Bti x Ai 1 x This second step is referred to as affine mapping Let s look at an example of how

the S Box computations work

Example 0 We assume the S Box input Ai 2 C2 hex From Ta ble we can see that the inverse is

Ai 1 Bti 2F hex 2

We now apply the Bti bit vector as input to the affine transformation Note that the least significant bit lsb bt0 of Bti is at the rightmost position

Thus S C2 hex 25 hex which is exactly the result that is also given in the S Box Table

If one computes both steps for all possible input elements of the S Box and stores the results one obtains Table In most AES implementations in particular in virtually all software realizations of AES the S Box outputs are not explicitly computed as shown here but rather lookup tables like Table are used However for hardware implementations it is sometimes advantageous to realize the S Boxes as digital circuits which actually compute the inverse followed by the affine map ping

The advantage of using inversion in GF 28 as the core function of the Byte

Substitution layer is that it provides a high degree of nonlinearity which in turn provides optimum protection against some of the strongest known analytical attacks The affine step destroys the algebraic structure of the Galois field which in turn is needed to prevent attacks that would exploit the finite field inversion

In AES the Diffusion layer consists of two sublayers the ShiftRows transformation and the MixColumn transformation We recall that diffusion is the spreading of the influence of individual bits over the entire state Unlike the nonlinear S Box the

diffusion layer performs a linear operation on state matrices A B i e DIFF A

DIFF B DIFF A B

The ShiftRows transformation cyclically shifts the second row of the state matrix by three bytes to the right the third row by two bytes to the right and the fourth row by one byte to the right The first row is not changed by the ShiftRows trans formation The purpose of the ShiftRows transformation is to increase the diffusion properties of AES If the input of the ShiftRows sublayer is given as a state matrix B B0 B1 B15

the output is the new state

one position left shift

two positions left shift

three positions left shift

The MixColumn step is a linear transformation which mixes each column of the state matrix Since every input byte influences four output bytes the MixColumn operation is the major diffusion element in AES The combination of the ShiftRows and MixColumn layer makes it possible that after only three rounds every byte of the state matrix depends on all 16 plaintext bytes

In the following we denote the 16 byte input state by B and the 16 byte output state by C

where B is the state after the ShiftRows operation as given in Expression

Now each 4 byte column is considered as a vector and multiplied by a fixed

4 matrix The matrix contains constant entries Multiplication and addition of the coefficients is done in GF 28 As an example we show how the first four output bytes are computed

The second column of output bytes C4 C5 C6 C7 is computed by multiplying the four input bytes B4 B9 B14 B3 by the same constant matrix and so on Fig ure shows which input bytes are used in each of the four MixColumn operations We discuss now the details of the vector matrix multiplication which forms the MixColum operations We recall that each state byte Ci and Bi is an 8 bit value representing an element from GF 28 All arithmetic involving the coefficients is done in this Galois field For the constants in the matrix a hexadecimal notation is used 01 refers to the GF 28 polynomial with the coefficients i e it is the element 1 of the Galois field 02 refers to the polynomial with the bit vector i e to the polynomial x and 03 refers to the polynomial with the bit

vector i e the Galois field element x 1

The additions in the vector matrix multiplication are GF 28 additions that is simple bitwise XORs of the respective bytes For the multiplication of the con stants we have to realize multiplications with the constants 01 02 and 03 These are quite efficient and in fact the three constants were chosen such that software implementation is easy Multiplication by 01 is multiplication by the identity and does not involve any explicit operation Multiplication by 02 and 03 can be done through table look up in two by 8 tables As an alternative multiplication by 02 can also be implemented as a multiplication by x which is a left shift by one bit and a modular reduction with P x x8 x4 x3 x 1 Similarly multiplication by 03 which represents the polynomial x 1 can be implemented by a left shift by one bit and addition of the original value followed by a modular reduction with P x

Example 1 We continue with our example from Sect and assume that the input state to the MixColumn layer is

B 25 25 25

In this special case only two multiplications in GF 28 have to be done These are 02 25 and 03 25 which can be computed in polynomial notation

02 25 x x5 x2 1

03 25 x 1 x5 x2 1

x6 x3 x x5 x2 1

x6 x5 x3 x2 x 1

Since both intermediate values have a degree smaller than 8 no modular reduction with P x is necessary

The output bytes of C result from the following addition in GF 28

5 x6 x5 x3 x2 x 1

where i 0 15 This leads to the output state C 25 25 25

The two inputs to the Key Addition layer are the current 16 byte state matrix and a subkey which also consists of 16 bytes bits The two inputs are combined through a bitwise XOR operation Note that the XOR operation is equal to addi tion in the Galois field GF 2 The subkeys are derived in the key schedule that is described below in Sect

The key schedule takes the original input key of length or bit and derives the subkeys used in AES Note that an XOR addition of a subkey is used both at the input and output of AES This process is sometimes referred to as key whitening The number of subkeys is equal to the number of rounds plus one due to the key needed for key whitening in the first key addition layer cf Fig Thus for the key length of bits the number of rounds is nr 10 and there are 11 subkeys each of bits The AES with a bit key requires 13 subkeys of length bits and AES with a bit key has 15 subkeys The AES subkeys are computed recursively i e in order to derive subkey ki subkey ki 1 must be known etc

The AES key schedule is word oriented where 1 word 32 bits Subkeys are stored in a key expansion array W that consists of words There are different key schedules for the three different AES key sizes of and bit which are all fairly similar We introduce the three key schedules in the following

Key Schedule for Bit Key AES

The ll subkeys are stored in a key expansion array with the elements W 0 W 43 The subkeys are computed as depicted in Fig The elements K0 K15 denote the bytes of the original AES key

First we note that the first subkey k0 is the original AES key i e the key is copied into the first four elements of the key array W The other array elements are

AES key schedule for bit key size

computed as follows As can be seen in the figure the leftmost word of a subkey

W 4i where i 1 10 is computed as

W 4i W 4 i 1 g W 4i 1

Here g is a nonlinear function with a four byte input and output The remaining three words of a subkey are computed recursively as

W 4i j W 4i j 1 W 4 i 1 j

where i 1 10 and j 1 2 3 The function g rotates its four input bytes performs a byte wise S Box substitution and adds a round coefficient RC to it The round coefficient is an element of the Galois field GF 28 i e an 8 bit value It is only added to the leftmost byte in the function g The round coefficients vary from round to round according to the following rule

The function g has two purposes First it adds nonlinearity to the key sched ule Second it removes symmetry in AES Both properties are necessary to thwart certain block cipher attacks

Key Schedule for Bit Key AES

AES with bit key has 12 rounds and thus 13 subkeys of bit each The subkeys require 52 words which are stored in the array elements W 0 W 51 The computation of the array elements is quite similar to the bit key case and is shown in Fig There are eight iterations of the key schedule Note that these key schedule iterations do not correspond to the 12 AES rounds Each iteration computes six new words of the subkey array W The subkey for the first AES round is formed by the array elements W 0 W 1 W 2 W 3 the second subkey by the elements W 4 W 5 W 6 W 7 and so on Eight round coefficients RC i are needed within the function g They are computed as in the bit case and range from RC 1 RC 8

Key Schedule for Bit Key AES

AES with bit key needs 15 subkeys The subkeys are stored in the 60 words W 0 W 59 The computation of the array elements is quite similar to the bit key case and is shown in Fig The key schedule has seven iterations where each iteration computes eight words for the subkeys Again note that these key schedule iterations do not correspond to the 14 AES rounds The subkey for the first AES round is formed by the array elements W 0 W 1 W 2 W 3 the second subkey by the elements W 4 W 5 W 6 W 7 and so on There are seven round coefficients RC 1 RC 7 within the function g needed that are computed as in the bit case This key schedule also has a function h with 4 byte input and output The function applies the S Box to all four input bytes

In general when implementing any of the key schedules two different ap proaches exist

Precomputation All subkeys are expanded first into the array W The encryption decryption of a plaintext ciphertext is executed afterwards This approach is often taken in PC and server implementations of AES where large pieces of data are

encrypted under one key Please note that this approach requires nr 1 16 bytes of memory e g 11 16 bytes if the key size is bits This is the reason

AES key schedule for bit key sizes

why such an implementation on a device with limited memory resources such as a smart card is sometimes not desireable

On the fly A new subkey is derived for every new round during the encryption decryption of a plaintext ciphertext Please note that when decrypting cipher texts the last subkey is XORed first with the ciphertext Therefore it is required to recursively derive all subkeys first and then start with the decryption of a ciphertext and the on the fly generation of subkeys As a result of this overhead the decryption of a ciphertext is always slightly slower than the encryption of a plaintext when the on the fly generation of subkeys is used

AES key schedule for bit key size

Because AES is not based on a Feistel network all layers must actually be in verted i e the Byte Substitution layer becomes the Inv Byte Substitution layer the ShiftRows layer becomes the Inv ShiftRows layer and the MixColumn layer becomes Inv MixColumn layer However as we will see it turns out that the inverse layer operations are fairly similar to the layer operations used for encryption In ad

dition the order of the subkeys is reversed i e we need a reversed key schedule A block diagram of the decryption function is shown in Fig

inverse of round nr 1

AES decryption block diagram

Since the last encryption round does not perform the MixColum operation the first decryption round also does not contain the corresponding inverse layer All other decryption rounds however contain all AES layers In the following we dis cuss the inverse layers of the general AES decryption round Fig 9 Since the

XOR operation is its own inverse the key addition layer in the decryption mode is the same as in the encryption mode it consists of a row of plain XOR gates

AES decryption round function 1 2 nr 1

Inverse MixColumn Sublayer

After the addition of the subkey the inverse MixColumn step is applied to the state again the exception is the first decryption round In order to reverse the MixCol umn operation the inverse of its matrix must be used The input is a 4 byte column of the State C which is multiplied by the inverse matrix The matrix contains constant entries Multiplication and addition of the coefficients is done in GF 28

The second column of output bytes B4 B5 B6 B7 is computed by multiplying the four input bytes C4 C5 C6 C7 by the same constant matrix and so on Each value

Bi and Ci is an element from GF 28 Also the constants are elements from GF 28 The notation for the constants is hexadecimal and is the same as was used for the MixColumn layer for example

0B 0B hex 2 x3 x 1

Additions in the vector matrix multiplication are bitwise XORs

Inverse ShiftRows Sublayer

In order to reverse the ShiftRows operation of the encryption algorithm we must shift the rows of the state matrix in the opposite direction The first row is not changed by the inverse ShiftRows transformation If the input of the ShiftRows sublayer is given as a state matrix B B0 B1 B15

the inverse ShiftRows sublayer yields the output

one position right shift

two positions right shift

three positions right shift

Inverse Byte Substitution Layer

The inverse S Box is used when decrypting a ciphertext Since the AES S Box is a bijective i e a one to one mapping it is possible to construct an inverse S Box such that

Ai S 1 Bi S 1 S Ai

where Ai and Bi are elements of the state matrix The entries of the inverse S Box are given in Table

For readers who are interested in the details of how the entries of inverse S Box are constructed we provide a derivation However for a functional understanding of AES the remainder of this section can be skipped In order to reverse the S Box substitution we first have to compute the inverse of the affine transformation For this each input byte Bi is considered an element of GF 28 The inverse affine transformation on each byte Bi is defined by

Inverse AES S Box Substitution values in hexadecimal notation for input byte xy

where b7 b0 is the bitwise vector representation of Bi x and b 7 b 0 the result after the inverse affine transformation

In the second step of the inverse S Box operation the Galois field inverse has to be reversed For this note that Ai Ai 1 1 This means that the inverse operation is reversed by computing the inverse again In our notation we thus have to compute

Ai B i 1 GF 28

with the fixed reduction polynomial P x x8 x4 x3 x 1 Again the zero ele ment is mapped to itself The vector Ai a7 a0 representing the field element a a1x a0 is the result of the substitution

Decryption Key Schedule

Since decryption round one needs the last subkey the second decryption round needs the second to last subkey and so on we need the subkey in reversed order as shown in Fig In practice this is mainly achieved by computing the entire key schedule first and storing all 11 13 or 15 subkeys depending on the number or

Implementation in Software and Hardware

rounds AES is using which in turn depends on the three key lengths supported by AES This precomputation adds usually a small latency to the decryption operation relative to encryption

Implementation in Software and Hardware

We briefly comment on the efficiency of the AES cipher with respect to software and hardware implementation

Unlike DES AES was designed such that an efficient software implementation is possible A straightforward implementation of AES which directly follows the data path description such as the description given in this chapter is well suited for 8 bit processors such as those found on smart cards but is not particularly efficient on 32 bit or 64 bit machines which are common in today s PCs In a na ve imple mentation all time critical functions Byte Substitution ShiftRows MixColumn operate on individual bytes Processing 1 byte per instruction is inefficient on mod ern 32 bit or 64 bit processors

However the Rijndael designers proposed a method which results in fast soft ware implementations The core idea is to merge all round functions except the rather trivial key addition into one table look up This results in four tables each of which consists of entries where each entry is 32 bits wide These tables are named a T Box Four table accesses yield 32 output bits of one round Hence one round can be computed with 16 table look ups On a GHz Intel processor a throughput of Mbit s or 50 MByte s is possible The fastest known imple mentation on a 64 bit Athlon CPU achieves a theoretical throughput of more than

Gbit s However conventional hard disc encryption tools with AES or an open source implementation of AES reach a perfomance of a few hundred Mbit s on similar platforms

Compared to DES AES requires more hardware resources for an implementation However due to the high integration density of modern integrated circuits AES can be implemented with very high throughputs in modern ASIC or FPGA field programmable gate array these are programmable hardware devices technol ogy Commercial AES ASICs can exceed throughputs of 10Gbit sec Through par allelization of AES encryption units on one chip the speed can be further increased It can be said that symmetric encryption with today s ciphers is extremely fast not only compared to asymmetric cryptosystems but also compared to other algorithms

needed in modern communication systems such as data compression or signal pro cessing schemes

Discussion and Further Reading

AES Algorithm and Security A detailed description of the design principles of AES can be found in 52 This book by the Rijndael inventors describes the design of the block cipher Recent research in context to AES can be found online in the AES Lounge 68 This website is a dissemination effort within ECRYPT the Net work of Excellence in Cryptology and is a rich resource of activities around AES It gives many links to further information and papers regarding implementation and theoretical aspects of AES

There is currently no analytical attack against AES known which has a com plexity less than a brute force attack An elegant algebraic description was found which in turn triggered speculations that this could lead to attacks Subse quent research showed that an attack is in fact not feasible By now the common assumption is that the approach will not threaten AES A good summary on alge braic attacks can be found in 43 In addition there have been proposals for many other attacks including square attack impossible differential attack or related key attack Again a good source for further references is the AES Lounge

The standard reference for the mathematics of finite fields is A very acces sible but brief introduction is also given in 19 The International Workshop on the Arithmetic of Finite Fields WAIFI a relatively new workshop series is concerned with both the applications and the theory of Galois fields

Implementation As mentioned in Sect in most software implementations on modern CPUs special lookup tables are being used T Boxes An early detailed de scription of the construction of T Boxes can be found in 51 Sect 5 A description of a high speed software implementation on modern 32 bit and 64 bit CPUs is given in The bit slicing technique which was developed in the context of DES is also applicable to AES and can lead to very fast code as shown in

A strong indication for the importance of AES was the recent introduction of special AES instructions by Intel in CPUs starting in The instructions allow these machines to compute the round operation particularly quickly

There is wealth of literature dealing with hardware implementation of AES A good introduction to the area of AES hardware architectures is given in 0 As an example of the variety of AES implementations reference 86 de scribes a very small FPGA implementation with Mbit s and a very fast pipelined FPGA implementation with 25Gbit s It is also possible to use the DSP blocks i e fast arithmetic units available on modern FPGAs for AES which can also yield throughputs beyond 50Mbit s 63 The basic idea in all high speed architectures is to process several plaintext blocks in parallel by means of pipelining On the other end of the performance spectrum are lightweight architectures which are optimized

for applications such as RFID The basic idea here is to serialize the data path i e one round is processed in several time steps Good references are 75 42

AES is a modern block cipher which supports three key lengths of and bit It provides excellent long term security against brute force attacks

AES has been studied intensively since the late s and no attacks have been found that are better than brute force

AES is not based on Feistel networks Its basic operations use Galois field arith metic and provide strong diffusion and confusion

AES is part of numerous open standards such as IPsec or TLS in addition to being the mandatory encryption algorithm for US government applications It seems likely that the cipher will be the dominant encryption algorithm for many years to come

AES is efficient in software and hardware

Since May 26 the AES Advanced Encryption Standard describes the official standard of the US government

The evolutionary history of AES differs from that of DES Briefly describe the differences of the AES history in comparison to DES

Outline the fundamental events of the developing process

What is the name of the algorithm that is known as AES

Who developed this algorithm

Which block sizes and key lengths are supported by this algorithm

For the AES algorithm some computations are done by Galois Fields GF With the following problems we practice some basic computations

Compute the multiplication and addition table for the prime field GF 7 A mul

tiplication table is a square here table which has as its rows and columns all field elements Its entries are the products of the field element at the corresponding row and column Note that the table is symmetric along the diagonal The addition table is completely analogous but contains the sums of field elements as entries

Generate the multiplication table for the extension field GF 23 for the case that the irreducible polynomial is P x x3 x 1 The multiplication table is in this case a table Remark You can do this manually or write a program for it

Addition in GF 24 Compute A x B x mod P x in GF 24 using the ir reducible polynomial P x x4 x 1 What is the influence of the choice of the reduction polynomial on the computation

A x x2 1 B x x3 x2 1

A x x2 1 B x x 1

Multiplication in GF 24 Compute A x B x mod P x in GF 24 using the irreducible polynomial P x x4 x 1 What is the influence of the choice of the reduction polynomial on the computation

A x x2 1 B x x3 x2 1

A x x2 1 B x x 1

x4 x 1 x7 x6 x3 x2

where the irreducible polynomial is the one used by AES P x x8 x4 x3 x 1 Note that Table contains a list of all multiplicative inverses for this field

We consider the field GF 24 with P x x4 x 1 being the irreducible poly nomial Find the inverses of A x x and B x x2 x You can find the inverses

either by trial and error i e brute force search or by applying the Euclidean algo rithm for polynomials However the Euclidean algorithm is only sketched in this chapter Verify your answer by multiplying the inverses you determined by A and B respectively

Find all irreducible polynomials

of degree 3 over GF 2

of degree 4 over GF 2

The best approach for doing this is to consider all polynomials of lower degree and check whether they are factors Please note that we only consider monic irreducible polynomials i e polynomials with the highest coefficient equal to one

We consider AES with bit block length and bit key length What is the output of the first round of AES if the plaintext consists of ones and the first subkey i e the first subkey also consists of ones You can write your final results in a rectangular array format if you wish

In the following we check the diffusion properties of AES after a sin gle round Let W w0 w1 w2 w3 0x 0x 0x

0x be the input in 32 bit chunks to a bit AES The subkeys for the

computation of the result of the first round of AES are W0 W7 with 32 bits each

W0 B7E W1 8AED W2 0xABF 8 W3 9CF C W4 0xA0FAFE17 W5 0x 2CB1 W6 0x W7 A6C

Use this book to figure out how the input is processed in the first round e g S Boxes For the solution you might also want to write a short computer program or use an existing one In any case indicate all intermediate steps for the computation of ShiftRows SubBytes and MixColumns

Compute the output of the first round of AES to the input W and the subkeys

Compute the output of the first round of AES for the case that all input bits are zero

How many output bits have changed Remark that we only consider a single round after every further round more output bits will be affected avalanche effect

The MixColumn transformation of AES consists of a matrix vector multipli cation in the field GF 28 with P x x8 x4 x3 x 1 Let b b b0 be one of the four input bytes to the vector matrix multiplication Each input byte is multiplied with the constants 01 02 and 03 Your task is to provide exact equa tions for computing those three constant multiplications We denote the result by d d d0

Equations for computing the 8 bits of d 01 b

Equations for computing the 8 bits of d 02 b

Equations for computing the 8 bits of d 03 b

Note The AES specification uses 01 to represent the polynomial 1 02 to rep resent the polynomial x and 03 to represent x 1

We now look at the gate or bit complexity of the MixColumn function using the results from problem 1 We recall from the discussion of stream ciphers that a 2 input XOR gate performs a GF 2 addition

How many 2 input XOR gates are required to perform one constant multiplica tion by 01 02 and 03 respectively in GF 28

What is the overall gate complexity of a hardware implementation of one matrix vector multiplication

What is the overall gate complexity of a hardware implementation of the entire Diffusion layer We assume permutations require no gates

We consider the first part of the ByteSub operation i e the Galois field inver sion

Using Table what is the inverse of the bytes 29 F3 and 01 where each byte is given in hexadecimal notation

Verify your answer by performing a GF 28 multiplication with your answer and

the input byte Note that you have to represent each byte first as polynomials in

GF 28 The MSB of each byte represents the x7 coefficient

Your task is to compute the S Box i e the ByteSub values for the input bytes 29 F3 and 01 where each byte is given in hexadecimal notation

First look up the inverses using Table to obtain values Bt Now perform the affine mapping by computing the matrix vector multiplication and addition

Verify your result using the S Box Table

What is the value of S 0

Derive the bit representation for the following round constants within the key schedule

The minimum key length for the AES algorithm is bit Assume that a special purpose hardware key search machine can test one key in 10 ns on one pro cessor The processors can be parallelized Assume further that one such processor costs 10 including overhead Note that both the processor speed and the prize are rather optimistic assumptions We assume also that Moore s Law holds according to which processor performance doubles every 18 months

How long do we have to wait until an AES key search machine can be built which breaks the algorithm on average in one week and which doesn t cost more than 1 million

For the following we assume AES with bit key length Furthermore let us assume an ASIC which can check 3 keys per second

If we use 0 such ICs in parallel how long does an average key search take Compare this period of time with the age of the universe approx years

Assume Moore s Law will still be valid for the next few years how many years do we have to wait until we can build a key search machine to perform an average key search of AES in 24 hours Again assume that we use 0 ICs in parallel

More About Block Ciphers

A block cipher is much more than just an encryption algorithm It can be used as a versatile building block with which a diverse set of cryptographic mechanisms can be realized For instance we can use them for building different types of block based encryption schemes and we can even use block ciphers for realizing stream ciphers The different ways of encryption are called modes of operation and are discussed in this chapter Block ciphers can also be used for constructing hash func tions message authentication codes which are also knowns as MACs or key estab lishment protocols all of which will be described in later chapters There are also other uses for block ciphers e g as pseudo random generators In addition to modes of operation this chapter also discusses two very useful techniques for increasing the security of block ciphers namely key whitening and multiple encryption

In this chapter you will learn

the most important modes of operation for block ciphers in practice

security pitfalls when using modes of operations

the principles of key whitening

why double encryption is not a good idea and the meet in the middle attack

C Paar J Pelzl Understanding Cryptography

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

Encryption with Block Ciphers Modes of Operation

In the previous chapters we introduced how DES 3DES and AES encrypt a block of data Of course in practice one wants typically to encrypt more than one single 8 byte or 16 byte block of plaintext e g when encrypting an e mail or a computer file There are several ways of encrypting long plaintexts with a block cipher We introduce several popular modes of operation in this chapter including

Electronic Code Book mode ECB

Cipher Block Chaining mode CBC

Cipher Feedback mode CFB

Output Feedback mode OFB

The latter three modes use the block cipher as a building block for a stream cipher All of the five modes have one goal They encrypt data and thus provide confi dentiality for a message sent from Alice to Bob In practice we often not only want to keep data confidential but Bob also wants to know whether the message is re ally coming from Alice This is called authentication and the Galois Counter mode GCM which we will also introduce is a mode of operation that lets the receiver Bob determine whether the message was really sent by the person he shares a key with Alice Moreover authentication also allows Bob to detect whether the cipher

text was altered during transmission More on authentication is found in 0

The ECB and CFB modes require that the length of the plaintext be an exact multiple of the block size of the cipher used e g a multiple of 16 bytes in the case of AES If the plaintext does not have this length it must be padded There are several ways of doing this padding in practice One possible padding method is to append a single 1 bit to the plaintext and then to append as many 0 bits as necessary to reach a multiple of the block length Should the plaintext be an exact multiple of the block length an extra block consisting only of padding bits is appended

Electronic Codebook Mode ECB

The Electronic Code Book ECB mode is the most straightforward way of encrypt ing a message In the following let ek xi denote the encryption of plaintext block xi with key k using some arbitrary block cipher Let ek 1 yi denote the decryption of ciphertext block yi with key k Let us assume that the block cipher encrypts de

crypts blocks of size b bits Messages which exceed b bits are partitioned into b bit blocks If the length of the message is not a multiple of b bits it must be padded to a multiple of b bits prior to encryption As shown in Fig in ECB mode each block is encrypted separately The block cipher can for instance be AES or 3DES Encryption and decryption in the ECB mode is formally described as follows

Encryption and decryption in ECB mode

It is straightforward to verify the correctness of the ECB mode

ek 1 yi ek 1 ek xi xi

The ECB mode has advantages Block synchronization between the encryption and decryption parties Alice and Bob is not necessary i e if the receiver does not receive all encrypted blocks due to transmission problems it is still possible to de crypt the received blocks Similarly bit errors e g caused by noisy transmission lines only affect the corresponding block but not succeeding blocks Also block ci phers operating in ECB mode can be parallelized e g one encryption unit encrypts or decrypts block 1 the next one block 2 and so on This is an advantage for high speed implementations but many other modes such as the CFB do not allow parallelization

However as is often the case in cryptography there are some unexpected weak nesses associated with the ECB mode which we will discuss in the following The main problem of the ECB mode is that it encrypts highly deterministically This means that identical plaintext blocks result in identical ciphertext blocks as long as the key does not change The ECB mode can be viewed as a gigantic code book hence the mode s name which maps every input to a certain output Of course if the key is changed the entire code book changes but as long as the key is static the book is fixed This has several undesirable consequences First an attacker recog nizes if the same message has been sent twice simply by looking at the ciphertext Deducing information from the ciphertext in this way is called traffic analysis For instance if there is a fixed header that always precedes a message the header always results in the same ciphertext From this an attacker can for instance learn when a new message has been sent Second plaintext blocks are encrypted independently of previous blocks If an attacker reorders the ciphertext blocks this might result in valid plaintext and the reordering might not be detected We demonstrate two simple attacks which exploit these weaknesses of the ECB mode

The ECB mode is susceptible to substitution attacks because once a particular plaintext to ciphertext block mapping xi yi is known a sequence of ciphertext

blocks can easily be manipulated We demonstrate how a substitution attack could work in the real world Imagine the following example of an electronic wire transfer betweens banks

Example Substitution attack against electronic bank transfer

Let s assume a protocol for wire transfers between banks Fig 2 There are five fields which specify a transfer the sending bank s ID and account number the re ceiving bank s ID and account number and the amount We assume now and this is a major simplification that each of the fields has exactly the size of the block cipher width e g 16 bytes in the case of AES Furthermore the encryption key be tween the two banks does not change too frequently Due to the nature of the ECB an attacker can exploit the deterministic nature of this mode of operation by simple substitution of the blocks The attack details are as follows

Example for a substitution attack against ECB encryption

The attacker Oscar opens one account at bank A and one at bank B

Oscar taps the encrypted line of the banking communication network

He sends 0 transfers from his account at bank A to his account at bank B repeatedly He observes the ciphertexts going through the communication net work Even though he cannot decipher the random looking ciphertext blocks he can check for ciphertext blocks that repeat After a while he can recognize the five blocks of his own transfer He now stores blocks 1 3 and 4 of these transfers These are the encrypted versions of the ID numbers of both banks as well as the encrypted version of his account at bank B

Recall that the two banks do not change the key too frequently This means that the same key is used for several other transfers between bank A and B By com paring blocks 1 and 3 of all subsequent messages with the ones he has stored Oscar recognizes all transfers that are made from some account at bank A to some account at bank B He now simply replaces block 4 which contains the receiving account number with the block 4 that he stored before This block contains Oscar s account number in encrypted form As a consequence all trans fers from some account of bank A to some account of bank B are redirected to go into Oscar s B account Note that bank B now has means of detecting that the block 4 has been replaced in some of the transfers it receives

Withdraw money from bank B quickly and fly to a country that has a relaxed attitude about the extradition of white collar criminals

What s interesting about this attack is that it works completely without attack ing the block cipher itself So even if we would use AES with a bit key and if

we would encrypt each block say times this would not prevent the attack It should be stressed however that this is not an attack that breaks the block cipher itself Messages that are unknown to Oscar still remain confidential He simply re placed parts of the ciphertext with some other previous ciphertexts This is called a violation of the integrity of the message There are available techniques for pre serving the integrity of a message namely message authentication codes MACs and digital signatures Both are widely used in practice to prevent such an attack and are introduced in Chaps 10 and 12 Also the Galois Counter mode which is described below is an encryption mode with a built in integrity check Note that this attack only works if the key between bank A and B is not changed too frequently This is another reason why frequent key freshness is a good idea

We now look at another problem posed by the ECB mode

Example Encryption of bitmaps in ECB mode

Figure clearly shows a major disadvantage of the ECB mode Identical plaintexts are mapped to identical ciphertexts In case of a simple bitmap the information text in the picture can still be read out from the encrypted picture even though we used AES with a bit key for encryption This is because the background consists of only a few different plaintext blocks which yields a fairly uniformly looking back ground in the ciphertext On the other hand all plaintext blocks which contain part of the letters result in random looking ciphertexts These random looking cipher texts are clearly distinguishable from the uniform background by the human eye

Image and encrypted image using AES with bit key in ECB mode

This weakness is similar to the attack of the substitution cipher that was intro duced in the first example In both cases statistical properties in the plaintext are preserved in the ciphertext Note that unlike an attack against the substitution cipher or the above banking transfer attack an attacker does not have to do anything in the case here The human eye automatically makes use of the statistical information

Both attacks above were examples of the weakness of a deterministic encryption scheme Thus it is usually preferable that different ciphertexts are produced every time we encrypt the same plaintext This behavior is called probabilistic encryp tion This can be achieved by introducing some form of randomness typically in form of an initialization vector IV The following modes of operation all encrypt probabilistically by means of an IV

Cipher Block Chaining Mode CBC

There are two main ideas behind the Cipher Block Chaining CBC mode First the encryption of all blocks are chained together such that ciphertext yi depends not only on block xi but on all previous plaintext blocks as well Second the encryption is randomized by using an initialization vector IV Here are the details of the CBC mode

The ciphertext yi which is the result of the encryption of plaintext block xi is fed back to the cipher input and XORed with the succeeding plaintext block xi 1 This XOR sum is then encrypted yielding the next ciphertext yi 1 which can then be used for encrypting xi 2 and so on This process is shown on the left hand side of Fig For the first plaintext block x1 there is no previous ciphertext For this an IV is added to the first plaintext which also allows us to make each CBC encryption nondeterministic Note that the first ciphertext y1 depends on plaintext x1 and the IV The second ciphertext depends on the IV x1 and x2 The third ciphertext y3 depends on the IV and x1 x2 x3 and so on The last ciphertext is a function of all plaintext blocks and the IV

Encryption and decryption in CBC mode

When decrypting a ciphertext block yi in CBC mode we have to reverse the two operations we have done on the encryption side First we have to reverse the block cipher encryption by applying the decryption function e 1 After this we have to

undo the XOR operation by again XORing the correct ciphertext block This can be expressed for general blocks yi as ek 1 yi xi yi 1 The right hand side of Fig shows this process Again if the first ciphertext block y1 is decrypted the

result must be XORed with the initialization vector IV to determine the plaintext block x1 i e x1 IV ek 1 y1 The entire process of encryption and decryption can be described as

We now verify the mode i e we show that the decryption actually reverses the encryption For the decryption of the first block y1 we obtain

d y1 ek 1 y1 IV ek 1 ek x1 IV IV x1 IV IV x1

For the decryption of all subsequent blocks yi i 2 we obtain

d yi ek 1 yi yi 1 ek 1 ek xi yi 1 yi 1 xi yi 1 yi 1 xi

If we choose a new IV every time we encrypt the CBC mode becomes a prob abilistic encryption scheme If we encrypt a string of blocks x1 xt once with a first IV and a second time with a different IV the two resulting ciphertext sequences look completely unrelated to each other for an attacker Note that we do not have to keep the IV secret However in most cases we want the IV to be a nonce i e a number used only once There are many different ways of generating and agreeing on initialization values In the simplest case a randomly chosen number is trans mitted in the clear between the two communication parties prior to the encrypted session Alternatively it is a counter value that is known to Alice and Bob and it is incremented every time a new session starts which requires that the counter value must be stored between sessions It could be derived from values such as Alice s and Bob s ID number e g their IP addresses together with the current time Also in order to strengthen any of these methods we can take a value as described above and ECB encrypt it once using the block cipher with the key known to Alice and Bob and use the resulting ciphertext as the IV There are some advanced attacks which also require that the IV is nonpredictable

It is instructive to discuss whether the substitution attack against the bank trans

fer that worked for the ECB mode is applicable to the CBC mode If the IV is properly chosen for every wire transfer the attack will not work at all since Os car will not recognize any patterns in the ciphertext If the IV is kept the same for several transfers he would recognize the transfers from his account at bank A to

his account at bank B However if he substitutes ciphertext block 4 which is his encrypted account number in other wire transfers going from bank A to B bank B would decrypt block 4 and 5 to some random value Even though money would not be redirected into Oscar s account it might be redirected to some other random account The amount would be a random value too This is obviously also highly undesirable for banks This example shows that even though Oscar cannot perform specific manipulations ciphertext alterations by him can cause random changes to the plaintext which can have major negative consequences Hence in many if not in most real world systems encryption itself is not sufficient we also have to protect the integrity of the message This can be achieved by message authentication codes MACs or digital signatures which are introduced in 2 The Galois Counter mode described below provides encryption and integrity check simultaneously

Output Feedback Mode OFB

In the Output Feedback OFB mode a block cipher is used to build a stream cipher encryption scheme This scheme is shown in Fig Note that in OFB mode the key stream is not generated bitwise but instead in a blockwise fashion The output of the cipher gives us b key stream bits where b is the width of the block cipher used with which we can encrypt b plaintext bits using the XOR operation

The idea behind the OFB mode is quite simple We start with encrypting an IV with a block cipher The cipher output gives us the first set of b key stream bits The next block of key stream bits is computed by feeding the previous cipher output back into the block cipher and encrypting it This process is repeated as shown in Fig

The OFB mode forms a synchronous stream cipher cf as the key stream does not depend on the plain or ciphertext In fact using the OFB mode is quite sim ilar to using a standard stream cipher such as RC4 or Trivium Since the OFB mode forms a stream cipher encryption and decryption are exactly the same operation As can be seen in the right hand part of Fig the receiver does not use the block

cipher in decryption mode e 1 to decrypt the ciphertext This is because the actual encryption is performed by the XOR function and in order to reverse it i e to de

crypt it we simply have to perform another XOR function on the receiver side This is in contrast to ECB and CBC mode where the data is actually being encrypted and decrypted by the block cipher

Encryption and decryption using the OFB scheme is as follows

Encryption and decryption in OFB mode

As a result of the use of an IV the OFB encryption is also nondeterministic hence encrypting the same plaintext twice results in different ciphertexts As in the case for the CBC mode the IV should be a nonce One advantage of the OFB mode is that the block cipher computations are independent of the plaintext Hence one can precompute one or several blocks si of key stream material

Cipher Feedback Mode CFB

The Cipher Feedback CFB mode also uses a block cipher as a building block for a stream cipher It is similar to the OFB mode but instead of feeding back the output of the block cipher the ciphertext is fed back Hence a somewhat more accurate term for this mode would have been Ciphertext Feedback mode As in the OFB mode the key stream is not generated bitwise but instead in a blockwise fashion The idea behind the CFB mode is as follows To generate the first key stream block s1 we encrypt an IV For all subsequent key stream blocks s2 s3 we encrypt the previous ciphertext This scheme is shown in Fig

Since the CFB mode forms a stream cipher encryption and decryption are exactly the same operation The CFB mode is an example of an asynchronous stream cipher cf since the stream cipher output is also a function of the ciphertext

The formal description of the CFB mode follows

Encryption and decryption in CFB mode

As a result of the use of an IV the CFB encryption is also nondeterministic hence encrypting the same plaintext twice results in different ciphertexts As in the case for the CBC and OFB modes the IV should be a nonce

A variant of the CFB mode can be used in situations where short plaintext blocks are to be encrypted Let s use the encryption of the link between a remote key board and a computer as an example The plaintexts generated by the keyboard are typically only 1 byte long e g an ASCII character In this case only 8 bits of the key stream are used for encryption it does not matter which ones we choose as they are all secure and the ciphertext also only consists of 1 byte The feedback of the ciphertext as input to the block cipher is a bit tricky The previous block cipher input is shifted by 8 bit positions to the left and the 8 least significant positions of the in put register are filled with the ciphertext byte This process repeats Of course this approach works not only for plaintext blocks of length 8 but for any lengths shorter than the cipher output

Another mode which uses a block cipher as a stream cipher is the Counter CTR mode As in the OFB and CFB modes the key stream is computed in a blockwise fashion The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block Figure shows the principle

We have to be careful how to initialize the input to the block cipher We must prevent using the same input value twice Otherwise if an attacker knows one of

Encryption and decryption in counter mode

the two plaintexts that were encrypted with the same input he can compute the key stream block and thus immediately decrypt the other ciphertext In order to achieve this uniqueness often the following approach is taken in practice Let s assume a block cipher with an input width of bits such as an AES First we choose an IV that is a nonce with a length smaller than the block length e g 96 bits The remaining 32 bits are then used by a counter with the value CTR which is initialized to zero For every block that is encrypted during the session the counter is incremented but the IV stays the same In this example the number of blocks we can encrypt without choosing a new IV is Since every block consists of 8 bytes a maximum of 8 bytes or about 32 Gigabytes can be encrypted before a new IV must be generated Here is a formal description of the Counter mode with a cipher input construction as just introduced

Please note that the string IV CTR1 does not have to be kept secret It can for instance be generated by Alice and sent to Bob together with the first ciphertext block The counter CTR can either be a regular integer counter or a slightly more complex function such as a maximum length LFSR

One might wonder why so many modes are needed One attractive feature of the Counter mode is that it can be parallelized because unlike the OFB or CFB mode it does not require any feedback For instance we can have two block cipher engines running in parallel where the first block cipher encrypts the counter value CTR1 and the other CTR2 at the same time When the two block cipher engines are finished the first engine encrypts the value CTR3 and the other one CTR4 and so on This scheme would allow us to encrypt at twice the data rate of a single implementation Of course we can have more than two block ciphers running in parallel increasing the speed up proportionally For applications with high throughput demands e g

in networks with data rates in the range of Gigabits per second encryption modes that can be parallelized are very desirable

Galois Counter Mode GCM

The Galois Counter Mode GCM is an encryption mode which also computes a message authentication code MAC A MAC provides a cryptographic check sum that is computed by the sender Alice and appended to the message Bob also computes a MAC from the message and checks whether his MAC is the same as the one computed by Alice This way Bob can make sure that 1 the message was really created by Alice and 2 that nobody tampered with the ciphertext during transmission These two properties are called message authentication and integrity respectively Much more about MACs is found in 2 We presented a slightly simplified version of the GCM mode in the following

GCM protects the confidentiality of the plaintext x by using an encryption in counter mode Additionally GCM protects not only the authenticity of the plaintext x but also the authenticity of a string AAD called additional authenticated data This authenticated data is in contrast to the plaintext left in clear in this mode of operation In practice the string AAD might include addresses and parameters in a network protocol

The GCM consists of an underlying block cipher and a Galois field multiplier with which the two GCM functions authenticated encryption and authenticated de cryption are realized The cipher needs to have a block size of bits such as AES On the sender side GCM encrypts data using the Counter Mode CTR followed by the computation of a MAC value For encryption first an initial counter is derived from an IV and a serial number Then the initial counter value is incremented and this value is encrypted and XORed with the first plaintext block For subsequent plaintexts the counter is incremented and then encrypted Note that the underlying block cipher is only used in encryption mode GCM allows for precomputation of the block cipher function if the initialization vector is known ahead of time

For authentication GCM performs a chained Galois field multiplication For ev ery plaintext xi an intermediate authentication parameter gi is derived gi is com puted as the XOR sum of the current ciphertext yi and gi and multiplied by the constant H The value H is a hash subkey which is generated by encryption of the all zero input with the block cipher All multiplications are in the bit Galois field GF with the irreducible polynomial P x x x7 x2 x 1 Since only one multiplication is required per block cipher encryption the GCM mode adds very little computational overhead to the encryption

Figure shows a diagram of the GCM

Basic authenticated encryption in Galois Counter mode

The receiver of the packet y1 yn T ADD decrypts the ciphertext by also applying the Counter mode To check the authenticity of the data the receiver also

computes an authentication tag T l using the received ciphertext and ADD as input He employs exactly the same steps as the sender If T and T l match the receiver is

assured that the cipertext and ADD were not manipulated in transit and that only the sender could have generated the message

Exhaustive Key Search Revisited

In Sect 1 we saw that given a plaintext ciphertext pair x1 y1 a DES key can be exhaustively searched using the simple algorithm

DESk x1 y1 i 0 1 1

For most other block ciphers however a key search is somewhat more complicated Somewhat surprisingly a brute force attack can produce false positive results i e keys ki are found that are not the one used for the encryption yet they perform a correct encryption in Eq The likelihood of this occurring is related to the relative size of the key space and the plaintext space

A brute force attack is still possible but several pairs of plaintext ciphertext are needed The length of the respective plaintext required to break the cipher with a brute force attack is referred to as unicity distance After trying every possible key there should be just one plaintext that makes sense

Let s first look why one pair x1 y1 might not be sufficient to identify the correct

key For illustration purposes we assume a cipher with a block width of 64 bit and a key size of 80 bit If we encrypt x1 under all possible keys we obtain cipher texts However there exist only different ones and thus some keys must map x1 to the same ciphertext If we run through all keys for a given plaintext ciphertext pair we find on average 4 keys that perform the mapping ek x1 y1 This estimation is valid since the encryption of a plaintext for a given key can be viewed as a random selection of a 64 bit ciphertext string The phenomenon of mul tiple paths between a given plaintext and ciphertext is depicted in Fig in which k i denote the keys that map x1 to y1 These keys can be considered key candidates

plaintext space ciphertext space

Multiple keys map between one plaintext and one ciphertext

Among the approximately key candidates k i is the correct one that was used by to perform the encryption Let s call this one the target key In order to identify the target key we need a second plaintext ciphertext pair x2 y2 Again there are about key candidates that map x2 to y2 One of them is the target key The other keys can be viewed as randomly drawn from the possible ones It is crucial to note that the target key must be present in both sets of key candidates To determine the effectiveness of a brute force attack the crucial question is now What is the likelihood that another false key is contained in both sets The answer is given by the following theorem

Returning to our example and assuming two plaintext ciphertext pairs the likeli hood of a false key k f that performs both encryptions ekf x1 y1 and ekf x2 y2 is

This value is so small that for almost all practical purposes it is sufficient to test two plaintext ciphertext pairs If the attacker chooses to test three pairs the likelihood of a false key decreases to 2 64 12 As we saw from this example the like

lihood of a false alarm decreases rapidly with the number t of plaintext ciphertext

pairs In practice typically we only need a few pairs

The theorem above is not only important if we consider an individual block ci pher but also if we perform multiple encryptions with a cipher This issue is ad dressed in the following section

Increasing the Security of Block Ciphers

In some situations we wish to increase the security of block ciphers e g if a ci pher such as DES is available in hardware or software for legacy reasons in a given application We discuss two general approaches to strengthen a cipher multiple en cryption and key whitening Multiple encryption i e encrypting a plaintext more than once is already a fundamental design principle of block ciphers since the round function is applied many times to the cipher Our intuition tells us that the security of a block cipher against both brute force and analytical attacks increases by performing multiple encryptions in a row Even though this is true in principle there are a few surprising facts For instance doing double encryption does very little to increase the brute force resistance over a single encryption We study this

counterintuitive fact in the next section Another very simple yet effective approach to increase the brute force resistance of block ciphers is called key whitening it is also discussed below

We note here that when using AES we already have three different security levels given by the key lengths of and bits Given that there are no realistic attacks known against AES with any of those key lengths there appears no reason to perform multiple encryption with AES for practical systems However for some selected older ciphers especially for DES multiple encryption can be a useful tool

Double Encryption and Meet in the Middle Attack

Let s assume a block cipher with a key length of bits For double encryption a plaintext x is first encrypted with a key kL and the resulting ciphertext is encrypted again using a second key kR This scheme is shown in Fig

ekl i x zL i zR j ekR j y

0 Double encryption and meet in the middle attack

A na ve brute force attack would require us to search through all possible com binations of both keys i e the effective key lengths would be 2 and an exhaustive key search would require 2 2 22 encryptions or decryptions However using the meet in the middle attack the key space is drastically reduced This is a divide and conquer attack in which Oscar first brute force attacks the encryption on the left hand side which requires 2 cipher operations and then the right encryption which again requires 2 operations If he succeeds with this attack the total com plexity is 2 2 2 1 This is barely more complex than a key search of a single encryption and of course is much less complex than performing 22 search operations

The attack has two phases In the first one the left encryption is brute forced and a lookup table is computed In the second phase the attacker tries to find a match in the table which reveals both encryption keys Here are the details of this approach

Phase I Table Computation For a given plaintext x1 compute a lookup table for all pairs kL i zL i where ekL i x1 zL i and i 1 2 2 These computations are symbolized by the left arrow in the figure The zL i are the intermediate values that occur in between the two encryptions This list should be ordered by the values of the zL i The number of entries in the table is 2 with each entry being n bits wide Note that one of the keys we used for encryption must be the correct target key but we still do not know which one it is

Phase II Key Matching In order to find the key we now decrypt y1 i e we perform the computations symbolized by the right arrow in the figure We select the first possible key kR 1 e g the all zero key and compute

We now check whether zR 1 is equal to any of the zL i values in the table which we computed in the first phase If it is not in the table we increment the key to kR 1 decrypt y1 again and check whether this value is in the table We continue until we have a match

We now have what is called a collision of two values i e zL i zR j This gives us two keys The value zL i is associated with the key kL i from the left encryption and kR j is the key we just tested from the right encryption This means there exists a key pair kL i kR j which performs the double encryption

ekR j ekL i x1 y1

As discussed in Sect there is a chance that this is not the target key pair we are looking for since there are most likely several possible key pairs that perform the mapping x1 y1 Hence we have to verify additional key candidates by en crypting several plaintext ciphertext pairs according to Eq If the verification fails for any of the pairs x1 y1 x2 y2 we go back to beginning of Phase II and increment the key kR again and continue with the search

Let s briefly discuss how many plaintext ciphertext pairs we will need to rule out faulty keys with a high likelihood With respect to multiple mappings between a plaintext and a ciphertext as depicted in Fig double encryption can be modeled as a cipher with 2 key bits and n block bits In practice one often has 2 n in which case we need several plaintext ciphertext pairs The theorem in Sect can easily be adopted to the case of multiple encryption which gives us a useful guideline about how many x y pairs should be available

Let s look at an example

Example As an example if we double encrypt with DES and choose to test three plaintext ciphertext pairs the likelihood of a faulty key pair surviving all three key tests is

Let us examine the computational complexity of the meet in the middle attack

In the first phase of the attack corresponding to the left arrow in the figure we per form 2 encryptions and store them in 2 memory locations In the second stage corresponding to the right arrow in the figure we perform a maximum of 2 decryp tions and table look ups We ignore multiple key tests at this stage The total cost for the meet in the middle attack is

number of encryptions and decryptions 2 2 2 1

number of storage locations 2

This compares to 2 encryptions or decryptions and essentially no storage cost in the case of a brute force attack against a single encryption Even though the storage requirements go up quite a bit the costs in computation and memory are still only proportional to 2 Thus it is widely believed that double encryption is not worth the effort Instead triple encryption should be used this method is described in the following section

Note that for a more exact complexity analysis of the meet in the middle attack we would also need take the cost of sorting the table entries in Phase I into account as well as the table look ups in Phase II For our purposes however we can ignore these additional costs

Compared to double encryption a much more secure approach is the encryption of a block of data three times in a row

y ek3 ek2 ek1 x

In practice often a variant of the triple encryption from above is used

y ek e 1 ek x

This type of triple encryption is sometimes referred to as encryption decryption encryption EDE The reason for this has nothing to do with security If k1 k2 the operation effectively performed is

which is single encryption Since it is sometimes desirable that one implementation can perform both triple encryption and single encryption i e in order to interoper ate with legacy systems EDE is a popular choice for triple encryption Moreover for a bit security it is sufficient to choose two different keys k1 and k2 and set k3 k1 in case of 3DES

Of course we can still perform a meet in the middle attack as shown in Fig

ekl i x zL1i zR m ekR1 j ekR2 m y

1 Triple encryption and sketch of a meet in the middle attack

Again we assume bits per key The problem for an attacker is that she has to compute a lookup table either after the first or after the second encryption In both cases the attacker has to compute two encryptions or decryptions in a row in order to reach the lookup table Here lies the cryptographic strength of triple encryption There are 22k possibilities to run through all possible keys of two encryptions or decryptions In the case of 3DES this forces an attacker to perform key tests which is entirely infeasible with current technology In summary the meet in the middle attack reduces the effective key length of triple encryption from 3 to 2 Because of this it is often said that the effective key length of triple DES is bits

as opposed to 3 56 bits which are actually used as input to the cipher

Using an extremely simple technique called key whitening it is possible to make block ciphers such as DES much more resistant against brute force attacks The basic scheme is shown in Fig

In addition to the regular cipher key k two whitening keys k1 and k2 are used to XOR mask the plaintext and ciphertext This process can be expressed as

2 Key whitening of a block cipher

It is important to stress that key whitening does not strengthen block ciphers against most analytical attacks such as linear and differential cryptanalysis This is in contrast to multiple encryption which often also increases the resistance to analytical attacks Hence key whitening is not a cure for inherently weak ciphers Its main application is ciphers that are relatively strong against analytical attacks but possess too short a key space The prime example of such a cipher is DES A variant of DES which uses key whitening is DESX In the case of DESX the key k2 is derived from k and k1 Please note that most modern block ciphers such as AES already apply key whitening internally by adding a subkey prior to the first round and after the last round

Let s now discuss the security of key whitening A na ve brute force attack against the scheme requires 2 2n search steps where is the bit length of the key and n the block size Using the meet in the middle attack introduced in Sect the computational load can be reduced to approximately 2 n steps plus storage of 2n data sets However if the adversary Oscar can collect 2m plaintext ciphertext pairs a more advanced attack exists with a computational complexity of

cipher operations Even though we do not introduce the attack here we ll briefly discuss its consequences if we apply key whitening to DES We assume that the at tacker knows 2m plaintext ciphertext pairs Note that the designer of a security sys tem can often control how many plaintext ciphertext are generated before a new key is established Thus the parameter m cannot be arbitrarily increased by the attacker Also since the number of known plaintexts grows exponentially with m values be yond say m 40 seem quite unrealistic As a practical example let s assume key whitening of DES and that Oscar can collect a maximum of plaintexts He now has to perform

Discussion and Further Reading

DES computations Given that with today s technology even DES operations re quire several days with special hardware performing encryptions is completely out of reach Note that the number of plaintexts which Oscar is not supposed to know in most circumstances corresponds to 32 GByte of data the collection of which is also a formidable task in most real world situations

A particular attractive feature of key whitening is that the additional computa tional load is negligible A typical block cipher implementation in software requires several hundred instructions for encrypting one input block In contrast a 64 bit XOR operation only takes 2 instructions on a 32 bit machine so that the perfor mance impact due to key whitening is in the range of 1 or less in most cases

Discussion and Further Reading

Modes of Operation After the AES selection process the US National Institute of Standards and Technology NIST supported the process of evaluating new modes of operations in a series of special publications and workshops Currently there are eight approved block cipher modes five for confidentiality ECB CBC CFB OFB CTR one for authentication CMAC and two combined modes for confi dentiality and authentication CCM GCM The modes are widely used in practice and are part of many standards e g for computer networks or banking

Other Applications for Block Ciphers The most important application of block ciphers in practice in addition to data encryption is Message Authentication Codes MACs which are discussed in 2 The schemes CBC MAC OMAC and PMAC are constructed with a block cipher Authenticated Encryption AE uses block ciphers to both encrypt and generate a MAC in order to provide confidentiality and authentication respectively In addition to the GCM introduced in this chapter other AE modes include the EAX mode OCB mode and GC mode

Another application is the Cryptographically Secure Pseudo Random Number Generators CSPRNG built from block ciphers In fact the stream cipher modes introduced in this chapter OFB CFB and CTR mode form CSPRNGs There are also standards such as 4 Appendix A which explicitly specify random number generators from block ciphers

Block ciphers can also be used to build cryptographic hash functions as dis cussed in 1

Extending Brute Force Attacks Even though there are no algorithmic shortcuts to brute force attacks there are methods which are efficient if several exhaustive key searches have to be performed Those methods are called time memory tradeoff at tacks TMTO The general idea is to encrypt a fixed plaintext under a large number of keys and to store certain intermediate results This is the precomputation phase which is typically at least as complex as a single brute force attack and which results in large lookup tables In the online phase a search through the tables takes place which is considerably faster than a brute force attack Thus after the precomputa

tion phase individual keys can be found much more quickly TMTO attacks were originally proposed by Hellman 91 and were improved with the introduction of distinguished points by Rivest More recently rainbow tables were proposed to further improve TMTO attacks A limiting factor of TMTO attacks in prac tice is that for each individual attack it is required that the same piece of known plaintext was encrypted e g a file header

Block Ciphers and Quantum Computers With the potential rise of quantum computers in the future the security of currently used crypto algorithms has to be reevaluated It should be noted that the possible existence of quantum computers in a few decades from now is hotly debated Whereas all popular existing asymmetric algorithms such as RSA are vulnerable to attacks using quantum computers symmetric algorithms are much more resilient A potential quantum computer us ing Grover s algorithm 87 would require only 2 n 2 steps in order to perform a complete key search on a cipher with a keyspace of 2n elements Hence key lengths of more than bit are required if resistance against quantum computer attacks is desired This observation was also the motivation for requiring the bit and bit key lengths for AES Interestingly it can be shown that there can be no quantum algorithm which performs such an attack more efficiently than Grover s algorithm 16

There are many different ways to encrypt with a block cipher Each mode of operation has some advantages and disadvantages

Several modes turn a block cipher into a stream cipher

There are modes that perform encryption together together with authentication i e a cryptographic checksum protects against message manipulation

The straightforward ECB mode has security weaknesses independent of the un derlying block cipher

The counter mode allows parallelization of encryption and is thus suited for high speed implementations

Double encryption with a given block cipher only marginally improves the resis tance against brute force attacks

Triple encryption with a given block cipher roughly doubles the key length Triple DES 3DES has an effective key length of bits

Key whitening enlarges the DES key length without much computational over head

Consider the storage of data in encrypted form in a large database using AES One record has a size of 16 bytes Assume that the records are not related to one another Which mode would be best suited and why

We consider known plaintext attacks on block ciphers by means of an exhaus tive key search where the key is k bits long The block length counts n bits with n k

How many plaintexts and ciphertexts are needed to successfully break a block cipher running in ECB mode How many steps are done in the worst case

Assume that the initialization vector IV for running the considered block cipher in CBC mode is known How many plaintexts and ciphertexts are now needed to break the cipher by performing an exhaustive key search How many steps need now maximally be done Briefly describe the attack

How many plaintexts and ciphertexts are necessary if you do not know the IV

Is breaking a block cipher in CBC mode by means of an exhaustive key search considerably more difficult than breaking an ECB mode block cipher

In a company all files which are sent on the network are automatically en crypted by using AES in CBC mode A fixed key is used and the IV is changed once per day The network encryption is file based so that the IV is used at the beginning of every file

You managed to spy out the fixed AES key but do not know the recent IV Today you were able to eavesdrop two different files one with unidentified content and one which is known to be an automatically generated temporary file and only contains the value 0xFF Briefly describe how it is possible to obtain the unknown initialization vector and how you are able to determine the content of the unknown file

Keeping the IV secret in OFB mode does not make an exhaustive key search more complex Describe how we can perform a brute force attack with unknown IV What are the requirements regarding plaintext and ciphertext

Describe how the OFB mode can be attacked if the IV is not different for each execution of the encryption operation

Propose an OFB mode scheme which encrypts one byte of plaintext at a time e g for encrypting key strokes from a remote keyboard The block cipher used is AES Perform one block cipher operation for every new plaintext byte Draw a block diagram of your scheme and pay particular attention to the bit lengths used in your diagram cf the descripton of a byte mode at the end of Sect

As is so often true in cryptography it is easy to weaken a seemingly strong scheme by small modifications Assume a variant of the OFB mode by which we only feed back the 8 most significant bits of the cipher output We use AES and fill the remaining input bits to the cipher with 0s

Draw a block diagram of the scheme

Why is this scheme weak if we encrypt moderately large blocks of plaintext say kByte What is the maximum number of known plaintexts an attacker needs to completely break the scheme

Let the feedback byte be denoted by FB Does the scheme become cryptograph ically stronger if we feedback the bit value FB FB FB to the input i e we copy the feedback byte 16 times and use it as AES input

In the text a variant of the CFB mode is proposed which encrypts individual bytes Draw a block diagram for this mode when using AES as block cipher Indicate the width in bit of each line in your diagram

We are using AES in counter mode for encrypting a hard disk with 1 TB of capacity What is the maximum length of the IV

Sometimes error propagation is an issue when choosing a mode of operation in practice In order to analyze the propagation of errors let us assume a bit error i e a substitution of a 0 bit by a 1 bit or vice versa in a ciphertext block yi

Assume an error occurs during the transmission in one block of ciphertext let s say yi Which cleartext blocks are affected on Bob s side when using the ECB mode

Again assume block yi contains an error introduced during transmission Which cleartext blocks are affected on Bob s side when using the CBC mode

Suppose there is an error in the cleartext xi on Alice s side Which cleartext blocks are affected on Bob s side when using the CBC mode

Assume a single bit error occurs in the transmission of a ciphertext character in 8 bit CFB mode How far does the error propagate Describe exactly how each block is affected

Prepare an overview of the effect of bit errors in a ciphertext block for the modes ECB CBC CFB OFB and CTR Differentiate between random bit errors and specific bit errors when decrypting yi

Besides simple bit errors the deletion or insertion of a bit yields even more severe effects since the synchronization of blocks is disrupted In most cases the decryption of subsequent blocks will be incorrect A special case is the CFB mode with a feedback width of 1 bit Show that the synchronization is automatically re stored after 1 steps where is the block size of the block cipher

We now analyze the security of DES double encryption 2DES by doing a cost estimate

2DES x DESK2 DESK1 x

First let us assume a pure key search without any memory usage For this pur pose the whole key space spanned by K1 and K2 has to be searched How much does a key search machine for breaking 2DES worst case in 1 week cost

In this case assume ASICs which can perform keys per second at a cost of

5 per IC Furthermore assume an overhead of 50 for building the key search machine

Let us now consider the meet in the middle or time memory tradeoff attack in which we can use memory Answer the following questions

How many entries have to be stored

How many bytes not bits have to be stored for each entry

How costly is a key search in one week Please note that the key space has to be searched before filling up the memory completely Then we can begin to search the key space of the second key Assume the same hardware for both key spaces

For a rough cost estimate assume the following costs for hard disk space

0 GByte where 1 GByte Byte

Assuming Moore s Law when do the costs move below 1 million

Imagine that aliens rather than abducting earthlings and performing strange experiments on them drop a computer on planet Earth that is particularly suited for AES key searches In fact it is so powerful that we can search through and key bits in a matter of days Provide guidelines for the number of plaintext ciphertext pairs the aliens need so that they can rule out false keys with a reasonable likelihood Remark Since the existence of both aliens and human built computers for such key lengths seem extremely unlikely at the time of writing this problem is pure science fiction

Given multiple plaintext ciphertext pairs your objective is to attack an en cryption scheme based upon multiple encryptions

You want to break an encryption system E which makes use of triple AES encryption e g block length n bit key size of k bit How many tuples xi yi with yi eK xi do you need to level down the probability of finding a key K which matches the condition yi eK xi for one particular i but fails for most other values of i a so called false positive to Pr Kl K 0

What is the maximum key size of a block cipher that you could still effectively

attack with an error probability of at most Pr Kl K 0 1 if this cipher always uses double encryption l 2 and has a block length of n 80

Estimate the success probability if you are provided with four plaintext ciphertext blocks which are double encrypted using AES n bits k bits Please justify your results

Note that this is a purely theoretical problem Key spaces of size and beyond can not be brute forced

3DES with three different keys can be broken with about 22k encryptions and 2k memory cells k 56 Design the corresponding attack How many pairs x y should be available so that the probability to determine an incorrect key triple k1 k2 k3 is sufficiently low

This is your chance to break a cryptosystem As we know by now cryptogra phy is a tricky business The following problem illustrates how easy it is to turn a strong scheme into a weak one with minor modifications

We saw in this chapter that key whitening is a good technique for strengthening block ciphers against brute force attacks We now look at the following variant of key whitening against DES which we ll call DESA

DESAk k1 x DESk x k1

Even though the method looks similar to key whitening it hardly adds to the se curity Your task is to show that breaking the scheme is roughly as difficult as a brute force attack against single DES Assume you have a few pairs of plaintext ciphertext

Introduction to Public Key Cryptography

Before we learn about the basics of public key cryptography let us recall that the term public key cryptography is used interchangeably with asymmetric cryptogra phy they both denote exactly the same thing and are used synonymously

As stated in symmetric cryptography has been used for at least years Public key cryptography on the other hand is quite new It was publicly introduced by Whitfield Diffie Martin Hellman and Ralph Merkle in Much more recently in British documents which were declassified revealed that the researchers James Ellis Clifford Cocks and Graham Williamson from the UK s Government Communications Headquarters GCHQ discovered and realized the principle of public key cryptography a few years earlier in However it is still being debated whether the government office fully recognized the far reaching consequences of public key cryptography for commercial security applications

In this chapter you will learn

A brief history of public key cryptography

The pros and cons of public key cryptography

Some number theoretical topics that are needed for understanding public key algorithms most importantly the extended Euclidean algorithm

C Paar J Pelzl Understanding Cryptography

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

Symmetric vs Asymmetric Cryptography

In this chapter we will see that asymmetric i e public key algorithms are very dif ferent from symmetric algorithms such as AES or DES Most public key algorithms are based on number theoretic functions This is quite different from symmetric ci phers where the goal is usually not to have a compact mathematical description between input and output Even though mathematical structures are often used for small blocks within symmetric ciphers for instance in the AES S Box this does not mean that the entire cipher forms a compact mathematical description

Symmetric Cryptography Revisited

In order to understand the principle of asymmetric cryptography let us first recall the basic symmetric encryption scheme in Fig

Principle of symmetric key encryption

Such a system is symmetric with respect to two properties

The same secret key is used for encryption and decryption

The encryption and decryption function are very similar in the case of DES they are essentially identical

There is a simple analogy for symmetric cryptography as shown in Fig Assume there is a safe with a strong lock Only Alice and Bob have a copy of the key for the lock The action of encrypting of a message can be viewed as putting the message in the safe In order to read i e decrypt the message Bob uses his key and opens the safe

Analogy for symmetric encryption a safe with one lock

Symmetric vs Asymmetric Cryptography

Modern symmetric algorithms such as AES or 3DES are very secure fast and are in widespread use However there are several shortcomings associated with symmetric key schemes as discussed below

Key Distribution Problem The key must be established between Alice and Bob using a secure channel Remember that the communication link for the message is not secure so sending the key over the channel directly which would be the most convenient way of transporting it can t be done

Number of Keys Even if we solve the key distribution problem we must poten tially deal with a very large number of keys If each pair of users needs a separate pair of keys in a network with n users there are

key pairs and every user has to store n 1 keys securely Even for mid size net works say a corporation with people this requires more than 4 million key pairs that must be generated and transported via secure channels More about this problem is found in Sect 3 There are smarter ways of dealing with keys in symmetric cryptography networks as detailed in Sect however those ap proaches have other problems such as a single point of failure

No Protection Against Cheating by Alice or Bob Alice and Bob have the same capabilities since they possess the same key As a consequence symmetric cryptog raphy cannot be used for applications where we would like to prevent cheating by either Alice or Bob as opposed to cheating by an outsider like Oscar For instance in e commerce applications it is often important to prove that Alice actually sent a certain message say an online order for a flat screen TV If we only use symmet ric cryptography and Alice changes her mind later she can always claim that Bob the vendor has falsely generated the electronic purchase order Preventing this is called nonrepudiation and can be achieved with asymmetric cryptography as dis cussed in Sect 1 Digital signatures which are introduced in 0 provide nonrepudiation

Analogy for public key encryption a safe with public lock for depositing a message and a secret lock for retrieving a message

Principles of Asymmetric Cryptography

In order to overcome these drawbacks Diffie Hellman and Merkle had a revolution ary proposal based on the following idea It is not necessary that the key possessed by the person who encrypts the message that s Alice in our example is secret The crucial part is that Bob the receiver can only decrypt using a secret key In order to realize such a system Bob publishes a public encryption key which is known to everyone Bob also has a matching secret key which is used for decryption Thus Bob s key k consists of two parts a public part kpub and a private one kpr

A simple analogy of such a system is shown in Fig This systems works quite similarly to the good old mailbox on the corner of a street Everyone can put a letter in the box i e encrypt but only a person with a private secret key can retrieve letters i e decrypt If we assume we have cryptosystems with such a functionality a basic protocol for public key encryption looks as shown in Fig

kpub kpr k

Basic protocol for public key encryption

By looking at that protocol you might argue that even though we can encrypt a message without a secret channel for key establishment we still cannot exchange a key if we want to encrypt with say AES However the protocol can easily be mod ified for this use What we have to do is to encrypt a symmetric key e g an AES key using the public key algorithm Once the symmetric key has been decrypted by Bob both parties can use it to encrypt and decrypt messages using symmetric ciphers Figure shows a basic key transport protocol where we use AES as the symmetric cipher for illustration purposes of course one can use any other sym metric algorithm in such a protocol The main advantage of the protocol in Fig over the protocol in Fig is that the payload is encrypted with a symmetric cipher which tends to be much faster than an asymmetric algorithm

From the discussion so far it looks as though asymmetric cryptography is a desirable tool for security applications The question remains how one can build public key algorithms In Chaps 7 8 and 9 we introduce most asymmetric schemes of practical relevance They are all built from one common principle the one way function The informal definition of it is as follows

kpub kpr

choose random k y ekpub k

encrypt message x z AESk x

Basic key transport protocol with AES as an example of a symmetric cipher

Obviously the adjectives easy and infeasible are not particularly exact In mathematical terms a function is easy to compute if it can be evaluated in polyno mial time i e its running time is a polynomial expression In order to be useful in practical crypto schemes the computation y f x should be sufficiently fast that

it does not lead to unacceptably slow execution times in an application The inverse computation x f 1 y should be so computationally intensive that it is not feasi ble to evaluate it in any reasonable time period say 00 years when using the

best known algorithm

There are two popular one way functions which are used in practical public key schemes The first is the integer factorization problem on which RSA is based Given two large primes it is easy to compute the product However it is very dif ficult to factor the resulting product In fact if each of the primes has or more decimal digits the resulting product cannot be factored even with thousands of PCs running for many years The other one way function that is used widely is the dis crete logarithm problem This is not quite as intuitive and is introduced in

Practical Aspects of Public Key Cryptography

Actual public key algorithms will be introduced in the next chapters since there is some mathematics we must study first However it is very interesting to look at the principal security functions of public key cryptography which we address in this section

As shown in the previous section public key schemes can be used for encryption of data It turns out that we can do many other previously unimaginable things with public key cryptography The main functions that they can provide are listed below

We note that identification and encryption can also be achieved with symmetric ciphers but they typically require much more effort with key management It looks as though public key schemes can provide all functions required by modern security protocols Even though this is true the major drawback in practice is that encryption of data is very computationally intensive or more colloquially extremely slow with public key algorithms Many block and stream ciphers can encrypt about one hundred to one thousand times faster than public key algorithms Thus somewhat ironically public key cryptography is rarely used for the actual encryption of data On the other hand symmetric algorithms are poor at providing nonrepudiation and key establishment functionality In order to use the best of both worlds most practi cal protocols are hybrid protocols which incorporate both symmetric and public key algorithms Examples include the SSL TLS potocol that is commonly used for se cure Web connections or IPsec the security part of the Internet communication protocol

The Remaining Problem Authenticity of Public Keys

From the discussion so far we ve seen that a major advantage of asymmetric schemes is that we can freely distribute public keys as shown in the protocols in Figs and However in practice things are a bit more tricky because we still have to assure the authenticity of public keys In other words Do we really know that a certain public key belongs to a certain person In practice this issue is often

solved with what is called certificates Roughly speaking certificates bind a public key to a certain identity This is a major issue in many security application e g when doing e commerce transactions on the Internet We discuss this topic in more detail in Sect 2

Another problem which is not as fundamental is that public key algorithms re quire very long keys resulting in slow execution times The issue of key lengths and security is discussed below

Important Public Key Algorithms

In the previous chapters we learned about some block ciphers DES and AES How ever there exist many other symmetric algorithms Several hundred algorithms have been proposed over the years and even though a lot were found not to be secure there exist many cryptographically strong ones as discussed in Sect The situa tion is quite different for asymmetric algorithms There are only three major fami lies of public key algorithms which are of practical relevance They can be classified based on their underlying computational problem

The first two families were proposed in the mid s and elliptic curves were proposed in the mid s There are no known attacks against any of the schemes if the parameters especially the operand and key lengths are chosen carefully Al gorithms belonging to each of the families will be introduced in Chaps 7 8 and

9 It is important to note that each of the three families can be used to provide the main public key mechanisms of key establishment nonrepudiation through digital signatures and encryption of data

In addition to the three families above there have been proposals for several other public key schemes They often lack cryptographic maturity i e it is not known how robust they are against mathematical attacks Multivariate quadratic

MQ or some lattice based schemes are examples of this Another common prob lem is that they have poor implementation characteristics like key lengths in the range of megabytes e g the McEliece cryptosystems However there are also some other schemes for instance hyperelliptic curve cryptosystems which are both as ef ficient and secure as the three established families shown above but which simply have not gained widespread adoption For most applications it is recommended to use public key schemes from the three established algorithm families

Key Lengths and Security Levels

All three of the established public key algorithm families are based on number theoretic functions One distinguishing feature of them is that they require arith metic with very long operands and keys Not surprisingly the longer the operands and keys the more secure the algorithms become In order to compare different algorithms one often considers the security level An algorithm is said to have a security level of n bit if the best known attack requires 2n steps This is a quite natural definition because symmetric algorithms with a security level of n have a key of length n bit The relationship between cryptographic strength and security is not as straightforward in the asymmetric case though Table shows recommended bit lengths for public key algorithms for the four security levels 80 and bit We see from the table that RSA like schemes and discrete logarithm schemes require very long operands and keys The key length of elliptic curve schemes is significantly smaller yet still twice as long as symmetric ciphers with the same cryptographic strength

Bit lengths of public key algorithms for different security levels

You may want to compare this table with the one given in Sect 2 which provides information about the security estimations of symmetric key algorithms In order to provide long term security i e security for a timespan of several decades a security level of bit should be chosen which requires fairly long keys for all three algorithm families

An undesired consequence of the long operands is that public key schemes are extremely arithmetically intensive As mentioned earlier it is not uncommon that one public operation say a digital signature is by orders of magnitude slower than the encryption of one block using AES or 3DES Moreover the computational

complexity of the three algorithm families grows roughly with the cube bit length As an example increasing the bit length from to in a given RSA signature generation software results in an execution that is 33 27 times slower On modern PCs execution times in the range of several 10 msec to a few msec are common which does not pose a problem for many applications However public key perfor mance can be a more serious bottleneck in constrained devices where small CPUs are prevalent e g mobile phones or smart cards or on network servers that have to compute many public key operations per second Chaps 7 8 and 9 introduce several techniques for implementing public key algorithms reasonably efficiently

Essential Number Theory for Public Key Algorithms

We will now study a few techniques from number theory which are essential for public key cryptography We introduce the Euclidean algorithm Euler s phi func tion as well as Fermat s Little Theorem and Euler s theorem All are important for asymmetric algorithms especially for understanding the RSA crypto scheme

We start with the problem of computing the greatest common divisor gcd The gcd of two positive integers r0 and r1 is denoted by

and is the largest positive number that divides both r0 and r1 For instance gcd 21 9

3 For small numbers the gcd is easy to calculate by factoring both numbers and finding the highest common factor

Example Let r0 84 and r1 30 Factoring yields

r0 84 2 2 3 7

The gcd is the product of all common prime factors

2 3 6 gcd 30 84

For the large numbers which occur in public key schemes however factoring

often is not possible and a more efficient algorithm is used for gcd computations the Euclidean algorithm The algorithm which is also referred to as Euclid s algorithm is based on the simple observation that

gcd r0 r1 gcd r0 r1 r1

where we assume that r0 r1 and that both numbers are positive integers This property can easily be proven Let gcd r0 r1 g Since g divides both r0 and r1 we can write r0 g x and r1 g y where x y and x and y are coprime integers i e they do not have common factors Moreover it is easy to show that x y and y are also coprime It follows from here that

gcd r0 r1 r1 gcd g x y g y g

Let s verify this property with the numbers from the previous example

Example Again let r0 84 and r1 30 We now look at the gcd of r0 r1

r0 r1 54 2 3 3 3

The largest common factor still is 2 3 6 gcd 30 54 gcd 30 84

It also follows immediately that we can apply the process iteratively

gcd r0 r1 gcd r0 r1 r1 gcd r0 r1 gcd r0 m r1 r1

as long as r0 m r1 0 The algorithm uses the fewest number of steps if we choose the maximum value for m This is the case if we compute

gcd r0 r1 gcd r0 mod r1 r1

Since the first term r0 mod r1 is smaller than the second term r1 we usually swap them

gcd r0 r1 gcd r1 r0 mod r1

The core observation from this process is that we can reduce the problem of finding the gcd of two given numbers to that of the gcd of two smaller numbers This process can be applied recursively until we obtain finally gcd rl 0 rl Since each iteration preserves the gcd of the previous iteration step it turns out that this final gcd is the gcd of the original problem i e

gcd r0 r1 gcd rl 0 rl

We first show some examples for finding the gcd using the Euclidean algorithm and then discuss the algorithm a bit more formally

Example Let r0 27 and r1 21 Fig gives us some feeling for the al gorithm by showing how the lengths of the parameters shrink in every iteration The shaded parts in the iteration are the new remainders r2 6 first iteration and r3 3 second iteration which form the input terms for the next iterations Note

that in the last iteration the remainder is r4 0 which indicates the termination of the algorithm

gcd 27 21 gcd 1 21 gcd 21 6

gcd 21 6 gcd 3 6 gcd 6 3

gcd 6 3 gcd 0 3 gcd 3 0 3

gcd 27 21 gcd 21 6 gcd 6 3 gcd 3 0 3

Example of the Euclidean algorithm for the input values r0 27 and r1 21

It is also helpful to look at the Euclidean algorithm with slightly larger numbers as happens in Example

Example Let r0 and r1 The gcd is then computed as

By now we should have an idea of Euclid s algorithm and we can give a more

formal description of the algorithm

Note that the algorithm terminates if a remainder with the value ri 0 is com puted The remainder computed in the previous iteration denoted by rl 1 is the gcd of the original problem

The Euclidean algorithm is very efficient even with the very long numbers typi cally used in public key cryptography The number of iterations is close to the num ber of digits of the input operands That means for instance that the number of iterations of a gcd involving bit numbers is times a constant Of course algorithms with a few thousand iterations can easily be executed on today s PCs making the algorithms very efficient in practice

Extended Euclidean Algorithm

So far we have seen that finding the gcd of two integers r0 and r1 can be done by recursively reducing the operands However it turns out that finding the gcd is not the main application of the Euclidean algorithm An extension of the algorithm allows us to compute modular inverses which is of major importance in public key cryptography In addition to computing the gcd the extended Euclidean algorithm EEA computes a linear combination of the form

gcd r0 r1 s r0 t r1

where s and t are integer coefficients This equation is often referred to as Diophan tine equation

The question now is how do we compute the two coefficients s and t The idea behind the algorithm is that we execute the standard Euclidean algorithm but we express the current remainder ri in every iteration as a linear combination of the form

ri sir0 tir1

If we succeed with this we end up in the last iteration with the equation

rl gcd r0 r1 slr0 tlr1 sr0 tr1

This means that the last coefficient sl is the coefficient s in Eq we are looking for and also tl t Let s look at an example

Example We consider the extended Euclidean algorithm with the same values as in the previous example r0 and r1 On the left hand side we compute the standard Euclidean algorithm i e we compute new remainders r2 r3 Also

we have to compute the integer quotient qi 1 in every iteration On the right hand side we compute the coefficients si and ti such that ri sir0 tir1 The coefficients

are always shown in brackets

The algorithm computed the three parameters gcd 7 s 13 and

t 42 The correctness can be verified by

gcd 7 73 01 9 2

You should carefully watch the algebraic steps taking place in the right column

of the example above In particular observe that the linear combination on the right hand side is always constructed with the help of the previous linear combinations We will now derive recursive formulae for computing si and ri in every iteration Assume we are in iteration with index i In the two previous iterations we computed the values

ri 2 si 2 r0 ti 2 r1

ri 1 si 1 r0 ti 1 r1

In the current iteration i we first compute the quotient qi 1 and the new remainder

ri from ri 1 and ri 2

ri 2 qi 1 ri 1 ri

This equation can be rewritten as

ri ri 2 qi 1 ri 1 Recall that our goal is to represent the new remainder ri as a linear combination of

r0 and r1 as shown in Eq The core step for achieving this happens now in

Eq we simply substitute ri 2 by Eq and ri 1 by Eq

ri si ti qi 1 si ti

If we rearrange the terms we obtain the desired result

ri si 2 qi 1si 1 r0 ti 2 qi 1ti 1 r1

Eq also gives us immediately the recursive formulae for computing si and

ti namely si si 2 qi 1si 1 and ti ti 2 qi 1ti 1 These recursions are valid

for index values i 2 Like any recursion we need starting values for s0 s1 t0 t1 These initial values which we derive in Problem 3 can be shown to be s0 1 s1 0 t0 0 t1 1

As mentioned above the main application of the EEA in asymmetric cryptog raphy is to compute the inverse modulo of an integer We already encountered this problem in the context of the affine cipher in For the affine cipher we were required to find the inverse of the key value a modulo 26 With the Euclidean algorithm this is straightforward Let s assume we want to compute the inverse of r1 mod r0 where r1 r0 Recall from Sect 2 that the inverse only exists if gcd r0 r1 1 Hence if we apply the EEA we obtain s r0 t r1 1 gcd r0 r1 Taking this equation modulo r0 we obtain

s 0 t r1 1 mod r0

r1 t 1 mod r0

Equation is exactly the definition of the inverse of r1 That means that t itself is the inverse of r1

Thus if we need to compute an inverse a 1 mod m we apply the EEA with the input parameters m and a The output value t that is computed is the inverse Let s

Example Our goal is to compute mod 67 The values 12 and 67 are rela tively prime i e gcd 67 12 1 If we apply the EEA we obtain the coefficients s

and t in gcd 67 12 1 s 67 t 12 Starting with the values r0 67 and r1 12 the algorithm proceeds as follows

This gives us the linear combination

5 67 28 12 1

As shown above the inverse of 12 follows from here as

This result can easily be verified

28 12 1 mod 67

Note that the s coefficient is not needed and is in practice often not computed

Please note also that the result of the algorithm can be a negative value for t The result is still correct however We have to compute t t r0 which is a valid operation since t t r0 mod r0

For completeness we show how the EEA can also be used for computing mul tiplicative inverses in Galois fields In modern cryptography this is mainly relevant for the derivation of the AES S Boxes and for elliptic curve public key algorithms The EEA can be used completely analogously with polynomials instead of inte gers If we want to compute an inverse in a finite field GF 2m the inputs to the algorithm are the field element A x and the irreducible polynomial P x The EEA computes the auxiliary polynomials s x and t x as well as the greatest common divisor gcd P x A x such that

s x P x t x A x gcd P x A x 1

Note that since P x is irreducible the gcd is always equal to 1 If we take the equation above and reduce both sides modulo P x it is straightforward to see that the auxiliary polynomial t x is equal to the inverse of A x

s x 0 t x A x 1 mod P x

t x A 1 x mod P x

We give at this point an example of the algorithm for the small field GF 23

Example We are looking for the inverse of A x x2 in the finite field GF 23 with P x x3 x 1 The initial values for the t x polynomial are t0 x 0 t1 x 1

Iteration ri 2 x qi 1 x ri 1 x ri x ti x

2 x3 x 1 x x2 x 1 t2 t0 q 0 x 1 x

3 x2 x x 1 x t3 t1 q 1 x x 1 x2

4 x 1 1 x 1 t4 t2 q x 1 1 x2

x x 1 0 Termination since r5 0

Note that polynomial coefficients are computed in GF 2 and since addition and multiplication are the same operations we can always replace a negative coefficient such as x by a positive one The new quotient and the new remainder that are computed in every iteration are shown in brackets above The polynomials ti x are computed according to the recursive formula that was used for computing the integers ti earlier in this section The EEA terminates if the remainder is 0 which is the case in the iteration with index 5 The inverse is now given as the last ti x value that was computed i e t4 x

A 1 x t x t4 x x2 x 1

Here is the check that t x is in fact the inverse of x2 where we use the properties that x3 x 1 mod P x and x4 x2 x mod P x

t4 x x2 x4 x3 x2

x2 x x 1 x2 mod P x

Note that in every iteration of the EEA one uses long division not shown above to determine the new quotient qi 1 x and the new remainder ri x

The inverse in was computed using the extended Euclidean algorithm

We now look at another tool that is useful for public key cryptosystems especially for RSA We consider the ring Zm i e the set of integers 0 1 m 1 We are

interested in the at the moment seemingly odd problem of knowing how many numbers in this set are relatively prime to m This quantity is given by Euler s phi function which is defined as follows

We first look at some examples and calculate Euler s phi function by actually counting all the integers in Zm which are relatively prime

Example Let m 6 The associated set is Z6 0 1 2 3 4 5

Since there are two numbers in the set which are relatively prime to 6 namely 1 and 5 the phi function takes the value 2 i e 6 2

Here is another example

Example Let m 5 The associated set is Z5 0 1 2 3 4

This time we have four numbers which are relatively prime to 5 hence 5 4

From the examples above we can guess that calculating Euler s phi function by running through all elements and computing the gcd is extremely slow if the num bers are large In fact computing Euler s phi function in this na ve way is com pletely out of reach for the large numbers occurring in public key cryptography Fortunately there exists a relation to calculate it much more easily if we know the factorization of m which is given in following theorem

Since the value of n i e the number of distinct prime factors is always quite small even for large numbers m evaluating the product symbol is computationally easy Let s look at an example where we calculate Euler s phi function using the relation

Example 0 Let m The factorization of in the canonical factorization form is

m 16 15 24 3 5 pe1 pe2 pe3

There are three distinct prime factors i e n 3 The value for Euler s phi functions follows then as

m 24 23 31 30 51 50 8 2 4 64

That means that 64 integers in the range 0 1 are coprime to m The alternative method which would have required to evaluate the gcd times would have been much slower even for this small number

It is important to stress that we need to know the factorization of m in order to calculate Euler s phi function quickly in this manner As we will see in the next chapter this property is at the heart of the RSA public key scheme Conversely if we know the factorization of a certain number we can compute Euler s phi function and decrypt the ciphertext If we do not know the factorization we cannot compute the phi function and hence cannot decrypt

Fermat s Little Theorem and Euler s Theorem

We describe next two theorems which are quite useful in public key crpytography We start with Fermat s Little Theorem 1 The theorem is helpful for primality testing and in many other aspects of public key cryptography The theorem gives a seem ingly surprising result if we do exponentiations modulo an integer

1 You should not confuse this with Fermat s Last Theorem one of the most famous number theoretical problems which was proved in the s after years

We note that arithmetic in finite fields GF p is done modulo p and hence the theorem holds for all integers a which are elements of a finite field GF p The theorem can be stated in the form

which is often useful in cryptography One application is the computation of the inverse in a finite field We can rewrite the equation as a ap 2 1 mod p This is exactly the definition of the multiplicative inverse Thus we immediately have a

way for inverting an integer a modulo a prime

a 1 ap 2 mod p

We note that this inversion method holds only if p is a prime Let s look at an example

Example 1 Let p 7 and a 2 We can compute the inverse of a as

ap 2 25 32 4 mod 7

This is easy to verify 2 4 1 mod 7

Performing the exponentiation in Eq is usually slower than using the extended Euclidean algorithm However there are situations where it is advantageous to use Fermat s Little Theorem e g on smart cards or other devices which have a hard ware accelerator for fast exponentiation anyway This is not uncommon because many public key algorithms require exponentiation as we will see in subsequent chapters

A generalization of Fermat s Little Theorem to any integer moduli i e moduli that are not necessarily primes is Euler s theorem

Since it works modulo m it is applicable to integer rings Zm We show now an example for Euler s theorem with small values

Example 2 Let m 12 and a 5 First we compute Euler s phi function of m

12 22 3 22 21 31 30 4 2 3 1 4

Now we can verify Euler s theorem

5 12 54 1 mod 12

It is easy to show that Fermat s Little Theorem is a special case of Euler s theorem If p is a prime it holds that p p1 p0 p 1 If we use this value for Euler s theorem we obtain a p ap 1 1 mod p which is exactly Fermat s

Discussion and Further Reading

Public Key Cryptography in General Asymmetric cryptography was introduced in the landmark paper by Whitfield Diffie and Martin Hellman 58 Ralph Merkle independently invented the concept of asymmetric cryptography but proposed an entirely different public key algorithm There are a few good accounts of the history of public key cryptography The treatment in 57 by Diffie is recommended Another good overview on public key cryptography is A very instructive and detailed history of elliptic curve cryptography including the relatively intense com petition between RSA and ECC during the s is described in More recent development in asymmetric cryptography is tracked by the Workshop on Public Key Cryptography PKC series

Modular Arithmetic With respect to the mathematics introduced in this chapter the introductory books on number theory recommended in Sect make good sources for further reading In practical terms the Extended Euclidean Algorithm EEA is the most crucial since virtually all implementations of public key schemes incorporate it especially modular inversion An important acceleration technique for the scheme is the binary EEA Its advantage over the standard EEA is that it replaces divisions by bit shifts This is in particular attractive for the very long num bers occurring in public key schemes

Alternative Public Key Algorithms In addition to the three established families of asymmetric schemes there exist several others First there are algorithms which have been broken or are believed to be insecure e g knapsack schemes Second there are generalizations of the established algorithms e g hyperelliptic curves algebraic varieties or non RSA factoring based schemes These schemes use the same one way function that is integer factorization or the discrete logarithm in certain groups Third there are asymmetric algorithms which are based on differ ent one way functions Four families of one way function are of particular interest hash based code based lattice based and multivariate quadratic MQ public key algorithms There are of course reasons why they are not as widely used today

In most cases they have either practical drawbacks such as very long keys some times in the range of several megabytes or the cryptographic strength is not well understood Since about there has been growing interest in the cryptographic community in such asymmetric schemes This is in part motivated by the fact that no quantum computing attacks are currently known against these four families of alternative asymmetric schemes This is in contrast to RSA discrete logarithm and elliptic curve schemes and their variants which are all vulnerable to attacks using quantum computers Even though it is not clear whether quantum computers will ever exist the most optimistic estimates state that they are still several decades away the alternative public key algorithms are at times collectively referred to as post quantum cryptography A recent book 18 and a new workshop series 36 35 provide more information about this area of active research

Public key algorithms have capabilities that symmetric ciphers don t have in particular digital signature and key establishment functions

Public key algorithms are computationally intensive a nice way of saying that they are slow and hence are poorly suited for bulk data encryption

Only three families of public key schemes are widely used This is considerably fewer than in the case of symmetric algorithms

The extended Euclidean algorithm allows us to compute modular inverses quickly which is important for almost all public key schemes

Euler s phi function gives us the number of elements smaller than an integer n that are relatively prime to n This is an important function for the RSA crypto scheme

As we have seen in this chapter public key cryptography can be used for en cryption and key exchange Furthermore it has some properties such as nonrepu diation which are not offered by secret key cryptography

So why do we still use symmetric cryptography in current applications

In this problem we want to compare the computational performance of sym metric and asymmetric algorithms Assume a fast public key library such as OpenSSL that can decrypt data at a rate of Kbit sec using the RSA al gorithm on a modern PC On the same machine AES can decrypt at a rate of 17 Mbit sec Assume we want to decrypt a movie stored on a DVD The movie requires 1 GByte of storage How long does decryption take with either algorithm

Assume a small company with employees A new security policy de mands encrypted message exchange with a symmetric cipher How many keys are required if you are to ensure a secret communication for every possible pair of communicating parties

The level of security in terms of the corresponding bit length directly influ ences the performance of the respective algorithm We now analyze the influence of increasing the security level on the runtime

Assume that a commercial Web server for an online shop can use either RSA or ECC for signature generation Furthermore assume that signature generation for RSA and ECC takes ms and ms respectively

Determine the increase in runtime for signature generation if the security level from RSA is increased from bit to bit

How does the runtime increase from bit to 60 bit

Determine these numbers for the respective security levels of ECC

Describe the difference between RSA and ECC when increasing the security level

Hint Recall that the computational complexity of both RSA and ECC grows with the cube of bit length You may want to use Table to determine the adequate bit length for ECC given the security level of RSA

Using the basic form of Euclid s algorithm compute the greatest common di visor of

For this problem use only a pocket calculator Show every iteration step of Euclid s algorithm i e don t write just the answer which is only a number Also for every gcd provide the chain of gcd computations i e

gcd r0 r1 gcd r1 r2

Using the extended Euclidean algorithm compute the greatest common divisor and the parameters s t of

For every problem check if sr0 t r1 gcd r0 r1 is actually fulfilled The rules are the same as above use a pocket calculator and show what happens in every iteration step

With the Euclidean algorithm we finally have an efficient algorithm for finding the multiplicative inverse in Zm that is much better than exhaustive search Find the inverses in Zm of the following elements a modulo m

a 7 m 26 affine cipher

Note that the inverses must again be elements in Zm and that you can easily verify your answers

Determine m for m 12 15 26 according to the definition Check for each positive integer n smaller m whether gcd n m 1 You do not have to apply Eu clid s algorithm

Develop formulae for m for the special cases when

m p q where p and q are primes This case is of great importance for the RSA cryptosystem Verify your formula for m 15 26 with the results from the previous problem

Compute the inverse a 1 mod n with Fermat s Theorem if applicable or Eu ler s Theorem

Verify that Euler s Theorem holds in Zm m 6 9 for all elements a for which gcd a m 1 Also verify that the theorem does not hold for elements a for which gcd a m 1

For the affine cipher in Chapter 1 the multiplicative inverse of an element modulo 26 can be found as

Derive this relationship by using Euler s Theorem

The extended Euclidean algorithm has the initial conditions s0 1 s1 0 t0 0 t1 1 Derive these conditions It is helpful to look at how the general iteration formula for the Euclidean algorithm was derived in this chapter

After Whitfield Diffie and Martin Hellman introduced public key cryptography in their landmark paper 58 a new branch of cryptography suddenly opened up As a consequence cryptologists started looking for methods with which public key encryption could be realized In Ronald Rivest Adi Shamir and Leonard Adleman cf Fig proposed a scheme which became the most widely used asymmetric cryptographic scheme RSA

An early picture of Adi Shamir Ron Rivest and Leonard Adleman reproduced with permission from Ron Rivest

In this chapter you will learn

Practical aspects of RSA such as computation of the parameters and fast en cryption and decryption

Implementational aspects

C Paar J Pelzl Understanding Cryptography

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

The RSA crypto scheme sometimes referred to as the Rivest Shamir Adleman al gorithm is currently the most widely used asymmetric cryptographic scheme even though elliptic curves and discrete logarithm schemes are gaining ground RSA was patented in the USA but not in the rest of the world until

There are many applications for RSA but in practice it is most often used for

encryption of small pieces of data especially for key transport

digital signatures which is discussed in 0 e g for digital certificates on the Internet

However it should be noted that RSA encryption is not meant to replace sym metric ciphers because it is several times slower than ciphers such as AES This is because of the many computations involved in performing RSA or any other public key algorithm as we learn later in this chapter Thus the main use of the encryption feature is to securely exchange a key for a symmetric cipher key trans port In practice RSA is often used together with a symmetric cipher such as AES where the symmetric cipher does the actual bulk data encryption

The underlying one way function of RSA is the integer factorization problem Multiplying two large primes is computationally easy in fact you can do it with paper and pencil but factoring the resulting product is very hard Euler s theorem Theorem 3 and Euler s phi function play important roles in RSA In the fol lowing we first describe how encryption decryption and key generation work then we talk about practical aspects of RSA

Encryption and Decryption

RSA encryption and decryption is done in the integer ring Zn and modular com putations play a central role Recall that rings and modular arithmetic in rings were introduced in Sect 2 RSA encrypts plaintexts x where we consider the bit string representing x to be an element in Zn 0 1 n 1 As a consequence the bi nary value of the plaintext x must be less than n The same holds for the ciphertext Encryption with the public key and decryption with the private key are as shown below

In practice x y n and d are very long numbers usually bit long or more The value e is sometimes referred to as encryption exponent or public exponent and the private key d is sometimes called decryption exponent or private exponent If Alice wants to send an encrypted message to Bob Alice needs to have his public key n e and Bob decrypts with his private key d We discuss in Sect how these three crucial parameters d e and n are generated

Even without knowing more details we can already state a few requirements for the RSA cryptosystem

Since an attacker has access to the public key it must be computationally infea sible to determine the private key d given the public key values e and n

Since x is only unique up to the size of the modulus n we cannot encrypt more than l bits with one RSA encryption where l is the bit length of n

It should be relatively easy to calculate xe mod n i e to encrypt and yd mod n i e to decrypt This means we need a method for fast exponentiation with very long numbers

For a given n there should be many private key public key pairs otherwise an attacker might be able to perform a brute force attack It turns out that this re quirement is easy to satisfy

Key Generation and Proof of Correctness

A distinctive feature of all asymmetric schemes is that there is a set up phase dur ing which the public and private key are computed Depending on the public key scheme key generation can be quite complex As a remark we note that key gener ation is usually not an issue for block or stream ciphers

Here are the steps involved in computing the public and private key for an RSA cryptosystem

The condition that gcd e n 1 ensures that the inverse of e exists modulo

n so that there is always a private key d

Two parts of the key generation are nontrivial Step 1 in which the two large primes are chosen as well as Steps 4 and 5 in which the public and private key are computed The prime generation of Step 1 is quite involved and is addressed in Sect The computation of the keys d and e can be done at once using the extended Euclidean algorithm EEA In practice one often starts by first selecting a public parameter e in the range 0 e n The value e must satisfy the condition gcd e n 1 We apply the EEA with the input parameters n and e and obtain the relationship

gcd n e s n t e

If gcd e n 1 we know that e is a valid public key Moreover we also know that the parameter t computed by the extended Euclidean algorithm is the inverse of e and thus

In case that e and n are not relatively prime we simply select a new value for e and repeat the process Note that the coefficient s of the EEA is not required for RSA and does not need to be computed

We now see how RSA works by presenting a simple example

Example Alice wants to send an encrypted message to Bob Bob first computes his RSA parameters in Steps He then sends Alice his public key Alice encrypts the message x 4 and sends the ciphertext y to Bob Bob decrypts y using his private key

message x choose p 3 and q 11

n 3 1 11 1 20

y xe 43 31 mod 33

yd 4 x mod 33

Note that the private and public exponents fulfill the condition e d 1 mod n

Practical RSA parameters are much much larger As can be seen from the RSA modulus n should be at least bit long which results in a bit length for p and q of Here is an example of RSA parameters for this bit length

p E0DFD A ACEBC EFAB C5A C CB 9CE4DE 9AA B FD 8CB EB BF CE DC8CE C h

q EBE0FCF 6FD F0D D 7AEE4B

A 8AEA30BD0BA D 8F CA99

n CF FDF BDBB1A ABB A 1FD AD12FC76DA C 8AC 7CA A 79AB 35 D AAAC E F C 0F 7E C2B 01FD 0EC8EAD4F 06F AE 0CBD

2 FEDE1A CF C CC0ABECF F A11ADFh

e E4CCF FF BE1D F 1EDB5 AB DD CA DB 96AE D EB B8 D 2ED 6DD E 5CF

8B EC4ED C75FA 8FE FB FF 4

3 5B 7 BD D C7EED8EE 3DC39h

d C 3EE7 D4FBFD 5D EBF 8 8F 4A 6F CE01EB8AD B

8BAABB CC03F 4EC 8AE9ED6D C7CC CC D CFE32DFC 07F AA6AE E CE6 C0DF E D D D 1F 7B 04BED81h

What is interesting is that the message x is first raised to the eth power during encryption and the result y is raised to the dth power in the decryption and the result of this is again equal to the message x Expressed as an equation this process is

x xe d xde x mod n

This is the essence of RSA We will now prove why the RSA scheme works

Proof We need to show that decryption is the inverse function of encryption

dkpr ekpub x x We start with the construction rule for the public and private key d e 1 mod n By definition of the modulo operator this is equivalent to

d e 1 t n

where t is some integer Inserting this expression in Eq

dkpr y xde x1 t n xt n x1 x n t x mod n

This means we have to prove that x x n t x mod n We use now Euler s The orem from Sect 3 which states that if gcd x n 1 then 1 x n mod n A minor generalization immediately follows

1 1t x n t mod n where t is any integer For the proof we distinguish two cases

First case gcd x n 1

Euler s Theorem holds here and we can insert Eq into

dkpr y x n t x 1 x x mod n q e d

This part of the proof establishes that decryption is actually the inverse func tion of encryption for plaintext values x which are relatively prime to the RSA modulus n We provide now the proof for the other case

Second case gcd x n gcd x p q 1

Since p and q are primes x must have one of them as a factor

x r p or x s q

where r s are integers such that r q and s p Without loss of generality we assume x r p from which follows that gcd x q 1 Euler s Theorem holds in the following form

1 1t x q t mod q

where t is any positive integer We now look at the term x n t again

x n t x q 1 p 1 t x q t p 1 1 p 1 1 mod q

Using the definition of the modulo operator this is equivalent to

x n t 1 u q

where u is some integer We multiply this equation by x

x x n t x x u q

x r p u q

x r u p q

x x n t x mod n

Inserting Eq into Eq yields the desired result

dkpr x n t x x mod n

If this proof seems somewhat lengthy please remember that the correctness of RSA is simply assured by Step 5 of the RSA key generation phase The proof be comes simpler by using the Chinese Remainder Theorem which we have not intro duced

Encryption and Decryption Fast Exponentiation

Unlike symmetric algorithms such as AES DES or stream ciphers public key al gorithms are based on arithmetic with very long numbers Unless we pay close attention to how to realize the necessary computations we can easily end up with schemes that are too slow for practical use If we look at RSA encryption and de cryption in Eqs and we see that both are based on modular exponentia tion We restate both operations here for convenience

x xe mod n encryption

x dkpr y yd mod n decryption

A straightforward way of exponentiation looks like this

where SQ denotes squaring and MU L multiplication Unfortunately the exponents e and d are in general very large numbers The exponents are typically chosen in the range of bit or even larger The public exponent e is sometimes chosen to be a small value but d is always very long Straightforward exponentiation as shown above would thus require around 4 or more multiplications Since the number of atoms in the visible universe is estimated to be around comput ing 4 multiplications to set up one secure session for our Web browser is not

too tempting The central question is whether there are considerably faster meth ods for exponentiation available The answer is luckily yes Otherwise we could forget about RSA and pretty much all other public key cryptosystems in use today since they all rely on exponentiation One such method is the square and multiply algorithm We first show a few illustrative examples with small numbers before pre senting the actual algorithm

Example Let s look at how many multiplications are required to compute the simple exponentiation x8 With the straightforward method

we need seven multiplications and squarings Alternatively we can do something faster

which requires only three squarings that are roughly as complex as a multiplication

This fast method works fine but is restricted to exponents that are powers of 2 i e values e and d of the form 2i Now the question is whether we can extend the method to arbitrary exponents Let us look at another example

Example This time we have the more general exponent 26 i e we want to compute x26 Again the na ve method would require 25 multiplications A faster way is as follows

This approach takes a total of six operations two multiplications and four squarings

Looking at the last example we see that we can achieve the desired result by performing two basic operations

squaring the current result

multiplying the current result by the base element x

In the example above we computed the sequence SQ MU L SQ SQ MU L SQ However we do not know the sequence in which the squarings and multiplications have to be performed for other exponents One solution is the square and multiply algorithm It provides a systematic way for finding the sequence in which we have to perform squarings and multiplications by x for computing xH Roughly speaking the algorithm works as follows

The algorithm is based on scanning the bit of the exponent from the left the most significant bit to the right the least significant bit In every iteration i e for every exponent bit the current result is squared If and only if the currently

scanned exponent bit has the value 1 a multiplication of the current result by

x is executed following the squaring

This seems like a simple if somewhat odd rule For better understanding let s revisit the example from above This time let s pay close attention to the exponent bits

Example We again consider the exponentiation x26 For the square and multiply algorithm the binary representation of the exponent is crucial

x26 x 02 x h h h

The algorithm scans the exponent bits starting on the left with h4 and ending with the rightmost bit h0

0 x x12 inital setting bit processed h4 1

1a x x2 x SQ bit processed h3

1b x2 x x3 x x12 x MUL since h3 1

2a x x6 x 2 x SQ bit processed h2

2b no MUL since h2 0

3a x x12 x 2 x 2 SQ bit processed h1

3b x12 x x13 x 2 x12 x 2 MUL since h1 1

4a x x26 x 2 2 x 02 SQ bit processed h0

4b no MUL since h0 0

To understand the algorithm it is helpful to closely observe how the binary rep resentation of the exponent evolves We see that the first basic operation squaring results in a left shift of the exponent with a 0 put in the rightmost position The other basic operation multiplication by x results in filling a 1 into the rightmost position of the exponent Compare how the highlighted exponents change from iteration to iteration

Here is the pseudo code for the square and multiply algorithm

The modulo reduction is applied after each multiplication and squaring operation in order to keep the intermediate results small It is helpful to compare this pseudo code with the verbal description of the algorithm above

We determine now the complexity of the square and multiply algorithm for an exponent H with a bit length of t 1 i e log2 H t 1 The number of squarings is independent of the actual value of H but the number of multiplications is equal to the Hamming weight i e the number of ones in its binary representation Thus we provide here the average number of multiplication denoted by MU L

Because the exponents used in cryptography have often good random properties assuming that half of their bits have the value one is often a valid approximation

Example How many operations are required on average for an exponentiation with a bit exponent

Straightforward exponentiation takes 4 0 multiplications That is com pletely impossible no matter what computer resources we might have at hand How ever the square and multiply algorithm requires only

squarings and multiplications on average This is an impressive example for the difference of an algorithm with linear complexity straightforward exponentiation and logarithmic complexity square and multiply algorithm Remember though that each of the individual squarings and multiplications involves bit numbers That means the number of integer operations on a CPU is much higher than but certainly doable on modern computers

Speed up Techniques for RSA

As we learned in Sect RSA involves exponentiation with very long numbers Even if the low level arithmetic involving modular multiplication and squaring as well as the square and multiply algorithm are implemented carefully performing a full RSA exponentiation with operands of length bit or beyond is computa tionally intensive Thus people have studied speed up techniques for RSA since its invention We introduce two of the most popular general acceleration techniques in the following

Fast Encryption with Short Public Exponents

A surprisingly simple and very powerful trick can be used when RSA operations with the public key e are concerned This is in practice encryption and as we ll learn later verification of an RSA digital signature In this situation the public key e can be chosen to be a very small value In practice the three values e 3 e 17 and e 1 are of particular importance The resulting complexities when using these public keys are given in Table

Complexity of RSA exponentiation with short public exponents

These complexities should be compared to the t multiplications and squarings that are required for exponents of full length Here t 1 is the bit length of the RSA modulus n i e log2 n t 1 We note that all three exponents listed above have a low Hamming weight i e number of ones in the binary representation This results in a particularly low number of operations for performing an exponentiation Interestingly RSA is still secure if such short exponents are being used Note that the private key d still has in general the full bit length t 1 even though e is short

An important consequence of the use of short public exponents is that encryption of a message and verification of an RSA signature is a very fast operation In fact for these two operations RSA is in almost all practical cases the fastest public key scheme available Unfortunately there is no such easy way to accelerate RSA when the private key d is involved i e for decryption and signature generation Hence these two operations tend to be slow Other public key algorithms in particular el liptic curves are often much faster for these two operations The following section shows how we can achieve a more moderate speed up when using the private expo nent d

Fast Decryption with the Chinese Remainder Theorem

We cannot choose a short private key without compromising the security for RSA If we were to select keys d as short as we did in the case of encryption in the section above an attacker could simply brute force all possible numbers up to a given bit length i e 50 bit But even if the numbers are larger say bit there are key recovery attacks In fact it can be shown that the private key must have a length of at least t bit where t is the bit length of the modulus n In practice e is often chosen short and d has full bit length What one does instead is to apply a method which is based on the Chinese Remainder Theorem CRT We do not introduce the CRT itself here but merely how it applies to accelerate RSA decryption and signature generation

Our goal is to perform the exponentiation xd mod n efficiently First we note that the party who possesses the private key also knows the primes p and q The basic idea of the CRT is that rather than doing arithmetic with one long modulus n we do two individual exponentiations modulo the two short primes p and q This is a type of transformation arithmetic Like any transform there are three steps transforming into the CRT domain computation in the CRT domain and inverse transformation of the result Those three steps are explained below

Transformation of the Input into the CRT Domain

We simply reduce the base element x modulo the two factors p and q of the modulus

n and obtain what is called the modular representation of x

xp x mod p xq x mod q

Exponentiation in the CRT Domain

With the reduced versions of x we perform the following two exponentiations

where the two new exponents are given by

Note that both exponents in the transform domain dp and dq are bounded by p and

q respectively The same holds for the transformed results yp and yq Since the two

primes are in practice chosen to have roughly the same bit length the two exponents as well as yp and yq have about half the bit length of n

Inverse Transformation into the Problem Domain

The remaining step is now to assemble the final result y from its modular represen tation yp yq This follows from the CRT and can be done as

y qcp yp p cq yq mod n where the coefficients cp and cq are computed as

cp q 1 mod p cq p 1 mod q

Since the primes change very infrequently for a given RSA implementation the two expressions in brackets in Eq can be precomputed After the precomputations the entire reverse transformation is achieved with merely two modular multiplica tions and one modular addition

Before we consider the complexity of RSA with CRT let s have a look at an example

Example Let the RSA parameters be given by

q 13 d e 1 mod

We now compute an RSA decryption for the ciphertext y 15 using the CRT i e the value yd 3 mod In the first step we compute the modular represen tation of y

In the second step we perform the exponentiation in the transform domain with the short exponents These are

Here are the exponentiations

xp yp 4 64 9 mod 11

xq yq 2 11 mod 13

In the last step we have to compute x from its modular representation xp xq For this we need the coefficients

cp 6 mod 11 cq 6 mod 13 The plaintext x follows now as

x qcp xp pcq xq mod n

x 13 11 1 mod

x mod

If you want to verify the result you can compute yd mod using the square and multiply algorithm

We will now establish the computational complexity of the CRT method If we look at the three steps involved in the CRT based exponentiation we conclude that for a practical complexity analysis the transformation and inverse transformation can be ignored since the operations involved are negligible compared to the actual exponentiations in the transform domain For convenience we restate these CRT exponentiations here

If we assume that n has t 1 bit both p and q are about t 2 bit long All numbers involved in the CRT exponentiations i e xp xq dp and dq are bound in size by p and q respectively and thus also have a length of about t 2 bit If we use the square and multiply algorithm for the two exponentiations each requires on average approximately t 2 modular multiplications and squarings Together the number of multiplications and squarings is thus

SQ MU L 2 t 2 t

This appears to be exactly the same computational complexity as regular exponen tiation without the CRT However each multiplication and squaring involves num bers which have a length of only t 2 bit This is in contrast to the operations without CRT where each multiplication was performed with t bit variables Since the com plexity of multiplication decreases quadratically with the bit length each t 2 bit multiplication is four times faster than a t bit multiplication 1 Thus the total speed up obtained through the CRT is a factor of 4 This speed up by four can be very valuable in practice Since there are hardly any drawbacks involved CRT based exponentiations are used in many cryptographic products e g for Web browser encryption The method is also particularly valuable for implementations on smart

1 The reason for the quadratic complexity is easy to see with the following example If we multiply a 4 digit decimal number abcd by another number wxyz we multiply each digit from the first operand with each digit of the second operand for a total of 42 16 digit multiplications On the other hand if we multiply two numbers with two digits i e ab times wx only 22 4 elementary multiplications are needed

cards e g for banking applications which are only equipped with a small micro processor Here digital signing is often needed which involves the secret key d By applying the CRT for signature computation the smart card is four times as fast For example if a regular bit RSA exponentiation takes 3 sec using the CRT reduces that time to 5 sec This acceleration might make the difference between a product with high customer acceptance 5 sec and a product with a delay that is not acceptable for many applications 3 sec This example is a good demonstration how basic number theory can have direct impact in the real world

There is one important practical aspect of RSA which we have not discussed yet generating the primes p and q in Step 1 of the key generation Since their product is the RSA modulus n p q the two primes should have about half the bit length of n For instance if we want to set up RSA with a modulus of length log2 n p and q should have a bit length of about bit The general approach is to generate integers at random which are then checked for primality as depicted in Fig where RNG stands for random number generator The RNG should be non predictable because if an attacker can compute or guess one of the two primes RSA can be broken easily as we will see later in this chapter

Principal approach to generating primes for RSA

In order to make this approach work we have to answer two questions

How many random integers do we have to test before we have a prime If the likelihood of a prime is too small it might take too long

How fast can we check whether a random integer is prime Again if the test is too slow the approach is impractical

It turns out that both steps are reasonably fast as is discussed in the following

How Common Are Primes

Now we ll answer the question whether the likelihood that a randomly picked inte ger p is a prime is sufficiently high We know from looking at the first few positive

integers that primes become less dense as the value increases

2 3 5 7 11 13 17 19 23 29 31 37

The question is whether there is still a reasonable chance that a random number with say bit is a prime Luckily this is the case The chance that a randomly picked integer p is a prime follows from the famous prime number theorem and is approximately 1 ln p In practice we only test odd numbers so that the likelihood doubles Thus the probability for a random odd number p to be prime is

In order to get a better feeling for what this probability means for RSA primes let s look at an example

Example For RSA with a bit modulus n the primes p and q each should have a length of about bits i e p q The probability that a random odd number p is a prime is

This means that we expect to test random numbers before we find one that is a prime

The likelihood of integers being primes decreases slowly proportional to the bit length of the integer This means that even for very long RSA parameters say with bit the density of primes is still sufficiently high

The other step we have to do is to decide whether the randomly generated integers p are primes A first idea could be to factor the number in question However for the numbers used in RSA factorization is not possible since p and q are too large In fact we especially choose numbers that cannot be factored because factoring n is the best known attack against RSA The situation is not hopeless though Remember that we are not interested in the factorization of p Instead we merely need the statement whether the number being tested is a prime or not It turns out that such primality tests are computationally much easier than factorization Examples for primality tests are the Fermat test the Miller Rabin test or variants of them We introduce primality test algorithms in this section

Practical primality tests behave somewhat unusually if the integer p in question is being fed into a primality test algorithm the answer is either

p is composite i e not a prime which is always a true statement or

p is prime which is only true with a high probability

If the algorithm output is composite the situation is clear The integer in question is not a prime and can be discarded If the output statement is prime p is probably a prime In rare cases however an integers prompts a prime statement but it lies i e it yields an incorrect positive answer There is way to deal with this behavior Practical primality tests are probabilistic algorithms That means they have a second parameter a as input which can be chosen at random If a composite number p together with a parameter a yields the incorrect statement p is prime we repeat the test a second time with a different value for a The general strategy is to test a

prime candidate p so often with several different random values a that the likelihood that the pair p a lies every single time is sufficiently small say less than 0 Remember that as soon as the statement p is composite occurs we know for

certain that p is not a prime and we can discard it

Fermat Primality Test

One primality test is based on Fermat s Little Theorem Theorem 2

The idea behind the test is that Fermat s theorem holds for all primes Hence if a number is found for which ap in Step it is certainly not a prime However the reverse is not true There could be composite numbers which in fact fulfill the condition ap In order to detect them the algorithm is run s times with different values of a

Unfortunately there are certain composite integers which behave like primes in the Fermat test for many values of a These are the Carmichael numbers Given a Carmichael number C the following expression holds for all integers a for which gcd a C 1

Such special composites are very rare For instance there exist approximately only Carmichael numbers below

Example Carmichael Number

n 3 11 17 is a Carmichael number since

for all gcd a 1

If the prime factors of a Carmichael numbers are all large there are only few bases a for which Fermat s test detects that the number is actually composite For this reason in practice the more powerful Miller Rabin test is often used to generate RSA primes

Miller Rabin Primality Test

In contrast to Fermat s test the Miller Rabin test does not have any composite num bers for which a large number of base elements a yield the statement prime The test is based on the following theorem

We can turn this into an efficient primality test

Step is computed by using the square and multiply algorithm The IF statement in Step tests the theorem for the case j 0 The FOR loop and the IF state

ment test the right hand side of the theorem for the values j 1 u 1

It can still happen that a composite number p gives the incorrect statement

prime However the likelihood of this rapidly decreases as we run the test with several different random base elements a The number of runs is given by the secu rity parameter s in the Miller Rabin test Table shows how many different values

a must be chosen in order to have a probability of less than 0 that a composite is incorrectly detected as a prime

Number of runs within the Miller Rabin primality test for an error probability of less than 0

Example Miller Rabin Test

Let p 91 Write p as p 1 5 We select a security parameter of s 4 Now choose s times a random value a

Let a 12 z 90 mod 91 hence p is likely prime

Let a 17 z 90 mod 91 hence p is likely prime

Let a 38 z 90 mod 91 hence p is likely prime

Let a 39 z 78 mod 91 hence p is composite

Since the numbers 12 17 and 38 give incorrect statements for the prime candidate

p 91 they are called liars for 91

RSA in Practice Padding

What we described so far is the so called schoolbook RSA system which has sev eral weaknesses In practice RSA has to be used with a padding scheme Padding schemes are extremely important and if not implemented properly an RSA imple mentation may be insecure The following properties of schoolbook RSA encryption are problematic

RSA encryption is deterministic i e for a specific key a particular plaintext is always mapped to a particular ciphertext An attacker can derive statistical properties of the plaintext from the ciphertext Furthermore given some pairs of plaintext ciphertext partial information can be derived from new ciphertexts which are encrypted with the same key

Plaintext values x 0 x 1 or x 1 produce ciphertexts equal to 0 1 or 1

Small public exponents e and small plaintexts x might be subject to attacks if no padding or weak padding is used However there is no known attack against small public exponents such as e 3

RSA has another undesirable property namely that it is malleable A crypto scheme is said to be malleable if the attacker Oscar is capable of transforming the ci phertext into another ciphertext which leads to a known transformation of the plain text Note that the attacker does not decrypt the ciphertext but is merely capable of manipulating the plaintext in a predictable manner This is easily achieved in the case of RSA if the attacker replaces the ciphertext y by se y where s is some integer If the receiver decrypts the manipulated ciphertext he computes

se y d sed xed sx mod n

Even though Oscar is not able to decrypt the ciphertext such targeted manipulations can still do harm For instance if x were an amount of money which is to be trans ferred or the value of a contract by choosing s 2 Oscar could exactly double the amount in a way that goes undetected by the receiver

A possible solution to all these problems is the use of padding which em beds a random structure into the plaintext before encryption and avoids the above mentioned problems Modern techniques such as Optimal Asymmetric Encryption Padding OAEP for padding RSA messages are specified and standardized in Pub lic Key Cryptography Standard 1 PKCS 1

Let M be the message to be padded let k be the length of the modulus n in bytes let H be the length of the hash function output in bytes and let M be the

RSA in Practice Padding

length of the message in bytes A hash function computes a message digest of fixed length e g or bit for every input More about hash functions is found in 1 Furthermore let L be an optional label associated with the message otherwise L is an empty string as default According to the most recent version PKCS 1 v padding a message within the RSA encryption scheme is done in the following way

Generate a string PS of length k M 2 H 2 of zeroed bytes The length of

Concatenate Hash L PS a single byte with hexadecimal value 1 and the message M to form a data block DB of length k H 1 bytes as

DB Hash L PS 1 M

Generate a random byte string seed of length H

Let dbMask MGF seed k H 1 where MGF is the mask generation func tion In practice a hash function such as SHA 1 is often used as MFG

Let maskedDB DB dbMask

Let seedMask MGF maskedDB H

Let maskedSeed seed seedMask

Concatenate a single byte with hexadecimal value 0 maskedSeed and

maskedDB to form an encoded message EM of length k bytes as

EM 0 maskedSeed maskedDB

Figure shows the structure of a padded message M

RSA encryption of a message M with Optimal Asymmetric Encryption Padding OAEP

On the decryption side the structure of the decrypted message has to be verified For instance if there is no byte with hexadecimal value 1 to separate PS from M a decryption error occurred In any case returning a decryption error to the user or a potential attacker should not reveal any information about the plaintext

There have been numerous attacks proposed against RSA since it was invented in None of the attacks are serious and moreover they typically exploit weak nesses in the way RSA is implemented or used rather than the RSA algorithm itself There are three general attack families against RSA

We comment on each of them in the following

Protocol attacks exploit weaknesses in the way RSA is being used There have been several protocol attacks over the years Among the better known ones are the attacks that exploit the malleability of RSA which was introduced in the previous section Many of them can be avoided by using padding Modern security standards describe exactly how RSA should be used and if one follows those guidelines protocol attacks should not be possible

The best mathematical cryptanalytical method we know is factoring the modulus An attacker Oscar knows the modulus n the public key e and the ciphertext y His goal is to compute the private key d which has the property that e d mod n It seems that he could simply apply the extended Euclidean algorithm and compute

d However he does not know the value of n At this point factoring comes in the best way to obtain this value is to decompose n into its primes p and q If Oscar can do this the attack succeeds in three steps

In order to prevent this attack the modulus must be sufficiently large This is the sole reason why moduli of or more bit are needed for a RSA The proposal of the RSA scheme in sparked much interest in the old problem of integer fac torization In fact the major progress that has been made in factorization in the last three decades would most likely not have happened if it weren t for RSA Table shows a summary of the RSA factoring records that have occurred since the begin ning of the s These advances have been possible mainly due to improvements in factoring algorithms and to a lesser extent due to improved computer technology

Even though factoring has become easier than the RSA designers had assumed 30 years ago factoring RSA moduli beyond a certain size still is out of reach

Summary of RSA factoring records since

Of historical interest is the digit modulus which was published in a column by Martin Gardner in Scientific American in It was estimated that the best factoring algorithms of that time would take 40 trillion 4 years However factoring methods improved considerably particularly during the s and s and it took in fact less than 30 years

Which exact length the RSA modulus should have is the topic of much discus sion Until recently many RSA applications used a bit length of bits as default Today it is believed that it might be possible to factor bit numbers within a pe riod of about 5 years and intelligence organizations might be capable of doing it possibly even earlier Hence it is recommended to choose RSA parameters in the range of bits for long term security

A third and entirely different family of attacks are side channel attacks They exploit information about the private key which is leaked through physical channels such as the power consumption or the timing behavior In order to observe such channels an attacker must typically have direct access to the RSA implementation e g in a cell phone or a smart card Even though side channel attacks are a large and active field of research in modern cryptography and beyond the scope of this book we show one particularly impressive such attack against RSA in the following

Figure shows the power trace of an RSA implementation on a microproces sor More precisely it shows the electric current drawn by the processor over time Our goal is to extract the private key d which is used during the RSA decryption We clearly see intervals of high activity between short periods of less activity Since the main computational load of RSA is the squarings and multiplication during the exponentiation we conclude that the high activity intervals correspond to those two operations If we look more closely at the power trace we see that there are high activity intervals which are short and others which are longer In fact the longer ones appear to be about twice as long This behavior is explained by the square and multiply algorithm If an exponent bit has the value 0 only a squaring is per

formed If an exponent bit has the value 1 a squaring together with a multiplication is computed But this timing behavior reveals immediately the key A long period of activity corresponds to the bit value 1 of the secret key and a short period to a key bit with value 0 As shown in the figure by simply looking at the power trace we can identify the secret exponent Thus we can learn the following 12 bits of the private key by looking at the trace

operations S SM SM S SM S S SM SM SM S SM

private key

Obviously in real life we can also find all or bits of a full private key During the short periods with low activity the square and multiply algorithm scans and processes the exponent bits before it triggers the next squaring or squaring and multiplication sequence

The power trace of an RSA implementation

This specific attack is classified as simple power analysis or SPA There are sev eral countermeasures available to prevent the attack A simple one is to execute a multiplication with dummy variables after a squaring that corresponds to an expo nent bit 0 This results in a power profile and a run time which is independent of the exponent However countermeasures against more advanced side channel at tacks are not as straightforward

Implementation in Software and Hardware

Implementation in Software and Hardware

RSA is the prime example almost literally for a public key algorithm that is very computationally intensive Hence the implementation of public key algorithms is much more crucial than that of symmetric ciphers like 3DES and AES which are significantly faster In order to get an appreciation for the computational load we develop a rough estimate for the number of integer multiplications needed for an RSA operation

We assume a bit RSA modulus For decryption we need on average squaring and multiplications each of which involves bit operands Let s as sume a 32 bit CPU so that each operand is represented by 32 64 registers A single long number multiplication requires now integer multiplica tions since we have to multiply every register of the first operand with every register of the second operand In addition we have to modulo reduce each of these multipli cations The best algorithms for doing this also require roughly integer multiplications Thus in total the CPU has to perform about integer multiplications for a single long number multiplication Since we have of these the number of integer multiplications for one decryption is

32 bit mult 25

Of course using a smaller modulus results in fewer operations but given that integer multiplications are among the most costly operations on current CPUs it is probably clear that the computational demand is quite impressive Note that most other public key schemes have a similar complexity

The extremely high computational demand of RSA was in fact a serious hin drance to its adoption in practice after it had been invented Doing hundreds of thousands of integer multiplications was out of question with s style comput ers The only option for RSA implementations with an acceptable run time was to realize RSA on special hardware chips until the mid to late s Even the RSA inventors investigated hardware architecture in the early days of the algorithm Since then much research has focused on ways to quickly perform modular integer arithmetic Given the enormous capabilities of state of the art VLSI chips an RSA operation can today be done in the range of s on high speed hardware

Similarly due to Moore s Law RSA implementations in software have become possible since the late s Today a typical decryption operation on a 2 GHz CPU takes around 10 ms for bit RSA Even though this is sufficient for many PC applications the throughput is about 48 bit s if one uses RSA for encryption of large amounts of data This is quite slow compared to the speed of many of today s networks For this reason RSA and other public key algorithms are not used for bulk data encryption Rather symmetric algorithms are used that are often faster by a factor of or so

Discussion and Further Reading

RSA and Variants The RSA cryptosystem is widely used in practice and is well standardized in bodies such as PKCS 1 Over the years several variants have been proposed One generalization is to use a modulus which is composed of more than two primes Also proposed have been multipower moduli of the form n p2 q

as well as multifactor ones where n pq r 45 In both cases speed ups by a factor of approximately are possible

There are also several other crypto schemes which are based on the integer fac torization problem A prominent one is the Rabin scheme In contrast to RSA it can be shown that the Rabin scheme is equivalent to factoring Thus it is said that the cryptosystem is provable secure Other schemes which rely on the hard ness of integer factorization include the probabilistic encryption scheme by Blum Goldwasser 28 and the Blum Blum Shub pseudo random number generator 27 The Handbook of Applied Cryptography describes all the schemes mentioned in a coherent form

Implementation The actual performance of an RSA implementation heavily de pends on the efficiency of the arithmetic used Generally speaking speed ups are possible at two levels On the higher level improvements of the square and multiply algorithm are an option One of the fastest methods is the sliding window exponen tiation which gives an improvement of about 25 over the square and multiply al gorithm A good compilation of exponentiation methods is given in 4 On the lower layer modular multiplication and squaring with long numbers can be improved One set of techniques deals with efficient algorithms for modular reduc tion In practice Montgomery reduction is the most popular choice see 41 for a good treatment of software techniques and 72 for hardware Several alternatives to the Montgomery method have also been proposed over the years 4 Another angle to accelerate long number arithmetic is to apply fast mul tiplication methods Spectral techniques such as the fast Fourier transform FFT are usually not applicable because the operands are still too short but methods such as the Karatsuba algorithm 99 are very useful Reference 17 gives a comprehensive but fairly mathematical treatment of the area of multiplication algorithms and describes the Karatsuba method from a practical viewpoint

Attacks Breaking RSA analytically has been a subject of intense investigation for the last 30 years Especially during the s major progress in factorization algo rithms was made which was not in small part motivated by RSA There have been numerous other attempts to mathematically break RSA including attacks against short private exponents A good survey is given in 32 More recently proposals have been made to build special computers whose sole purpose is to break RSA Proposals include an optoelectronic factoring machine and several other ar chitectures based on conventional semiconductor technology 79

Side channel attacks have been systematically studied in academia and industry since the mid to late s RSA as well as most other symmetric and asymmetric schemes are vulnerable against differential power analysis DPA which is more

powerful than the simple power analysis SPA shown in this section On the other hand numerous countermeasures against DPA are known Good references are The Side Channel Cryptanalysis Lounge 70 and the excellent book on DPA Related implementation based attacks are fault injection attacks and timing attacks It is important to stress that a cryptosystem can be mathematically very strong but still be vulnerable to side channel attacks

RSA is the most widely used public key cryptosystem In the future elliptic curve cryptosystems will probably catch up in popularity

RSA is mainly used for key transport i e encryption of keys and digital signa tures

The public key e can be a short integer The private key d needs to have the full length of the modulus Hence encryption can be significantly faster than decryption

RSA relies on the integer factorization problem Currently bit about decimal digits numbers cannot be factored Progress in factorization algorithms and factorization hardware is hard to predict It is advisable to use RSA with a bit modulus if one needs reasonable long term security especially with respect to extremely well funded attackers

Schoolbook RSA allows several attacks and in practice RSA should be used together with padding

Let the two primes p 41 and q 17 be given as set up parameters for RSA

Which of the parameters e1 32 e2 49 is a valid RSA exponent Justify your choice

Compute the corresponding private key Kpr p q d Use the extended Eu

clidean algorithm for the inversion and point out every calculation step

Computing modular exponentiation efficiently is inevitable for the practicabil ity of RSA Compute the following exponentiations xe mod m applying the square and multiply algorithm

After every iteration step show the exponent of the intermediate result in binary notation

Encrypt and decrypt by means of the RSA algorithm with the following system parameters

p 3 q 11 d 7 x 5

p 5 q 11 e 3 x 9

Only use a pocket calculator at this stage

One major drawback of public key algorithms is that they are relatively slow In Sect we learned that an acceleration technique is to use short exponents e Now we study short exponents in this problem in more detail

Assume that in an implementation of the RSA cryptosystem one modular squar ing takes 75 of the time of a modular multiplication How much quicker is one encryption on average if instead of a bit public key the short exponent e 1 is used Assume that the square and multiply algorithm is being used in both cases

Most short exponents are of the form e 2n 1 Would it be advantageous to use exponents of the form 2n 1 Justify your answer

Compute the exponentiation xe mod 29 of x 5 with both variants of e from

above for n 4 Use the square and multiply algorithm and show each step of your computation

In practice the short exponents e 3 17 and 1 are widely used

Why can t we use these three short exponents as values for the exponent d in applications where we want to accelerate decryption

Suggest a minimum bit length for the exponent d and explain your answer

Verify the RSA with CRT example in the chapter by computing yd 3 mod using the square and multiply algorithm

An RSA encryption scheme has the set up parameters p 31 and q 37 The public key is e 17

Decrypt the ciphertext y 2 using the CRT

Verify your result by encrypting the plaintext without using the CRT

Popular RSA modulus sizes are and bit

How many random odd integers do we have to test on average until we expect to find one that is a prime

Derive a simple formula for any arbitrary RSA modulus size

One of the most attractive applications of public key algorithms is the estab lishment of a secure session key for a private key algorithm such as AES over an insecure channel

Assume Bob has a pair of public private keys for the RSA cryptosystem Develop a simple protocol using RSA which allows the two parties Alice and Bob to agree on a shared secret key Who determines the key in this protocol Alice Bob or both

In practice it is sometimes desirable that both communication parties influ ence the selection of the session key For instance this prevents the other party from choosing a key which is a weak key for a symmetric algorithm Many block ciphers such as DES and IDEA have weak keys Messages encrypted with weak keys can be recovered relatively easily from the ciphertext

Develop a protocol similar to the one above in which both parties influence the key Assume that both Alice and Bob have a pair of public private keys for the RSA cryptosystem Please note that there are several valid approaches to this problem Show just one

In this exercise you are asked to attack an RSA encrypted message Imagine being the attacker You obtain the ciphertext y by eavesdropping on a certain connection The public key is kpub n e

Consider the encryption formula All variables except the plaintext x are known Why can t you simply solve the equation for x

In order to determine the private key d you have to calculate d e 1 mod n There is an efficient expression for calculating n Can we use this formula

Calculate the plaintext x by computing the private key d through factoring n p q Does this approach remain suitable for numbers with a length of bit or more

We now show how an attack with chosen ciphertext can be used to break an RSA encryption

Show that the multiplicative property holds for RSA i e show that the product of two ciphertexts is equal to the encryption of the product of the two respective plaintexts

This property can under certain circumstances lead to an attack Assume that Bob first receives an encrypted message y1 from Alice which Oscar obtains by eavesdropping At a later point in time we assume that Oscar can send an inno cent looking ciphertext y2 to Bob and that Oscar can obtain the decryption of y2 In practice this could for instance happen if Oscar manages to hack into Bob s system such that he can get access to decrypted plaintext for a limited period of time

In this exercise we illustrate the problem of using nonprobabilistic cryptosys tems such as schoolbook RSA imprudently Nonprobabilistic means that the same sequence of plaintext letters maps to the same ciphertext This allows traffic analysis i e to draw some conclusion about the cleartext by merely observing the cipher text and in some cases even to the total break of the cryptoystem The latter holds especially if the number of possible plaintexts is small Suppose the following situ ation

Alice wants to send a message to Bob encrypted with his public key pair n e

Therefore she decides to use the ASCII table to assign a number to each character Space 32 33 A 65 B 66 and to encrypt them separately

Oscar eavesdrops on the transferred ciphertext Describe how he can successfully decrypt the message by exploiting the nonprobabilistic property of RSA

Bob s RSA public key is n e 11 Decrypt the ciphertext

y

with the attack proposed in 1 For simplification assume that Alice only chose capital letters A Z during the encryption

Is the attack still possible if we use the OAEP padding Exactly explain your answer

The modulus of RSA has been enlarged over the years in order to thwart im proved attacks As one would assume public key algorithms become slower as the modulus length increases We study the relation between modulus length and perfor mance in this problem The performance of RSA and of almost any other public key algorithm is dependent on how fast modulo exponentiation with large numbers can be performed

Assume that one modulo multiplication or squaring with k bit numbers takes c k2 clock cycles where c is a constant How much slower is RSA encryp tion decryption with bits compared to RSA with bits on average Only consider the encryption decryption itself with an exponent of full length and the square and multiply algorithm

In practice the Karatsuba algorithm which has an asymptotical complexity that is proportional to klog is often used for long number multiplication in cryptog raphy Assume that this more advanced technique requires cl klog cl k 85

clock cycles for multiplication or squaring where cl is a constant What is the

ratio between RSA encryption with bit and RSA with bit if the Karat suba algorithm is used in both cases Again assume that full length exponents are being used

Advanced problem There are ways to improve the square and multiply al gorithm that is to reduce the number of operations required Although the number of squarings is fixed the number of multiplications can be reduced Your task is to come up with a modified version of the square and multiply algorithm which re quires fewer multiplications Give a detailed description of how the new algorithm works and what the complexity is number of operations

Hint Try to develop a generalization of the square and multiply algorithm which processes more than one bit at a time The basic idea is to handle k e g k 3 exponent bit per iteration rather than one bit in the original square and multiply algorithm

Let us now investigate side channel attacks against RSA In a simple imple mentation of RSA without any countermeasures against side channel leakage the analysis of the current consumption of the microcontroller in the decryption part directly yields the private exponent Figure shows the power consumption of an implementation of the square and multiply algorithm If the microcontroller com putes a squaring or a multiplication the power consumption increases Due to the small intervals in between the loops every iteration can be identified Furthermore for each round we can identify whether a single squaring short duration or a squar ing followed by a multiplication long duration is being computed

Identify the respective rounds in the figure and mark these with S for squaring or SM for squaring and multiplication

Assume the square and multiply algorithm has been implemented such that the exponent is being scanned from left to right Furthermore assume that the start ing values have been initialized What is the private exponent d

This key belongs to the RSA setup with the primes p 67 and q and

e Verify your result Note that in practice an attacker wouldn t know the values of p and q

Power consumption of an RSA decryption

Public Key Cryptosystems Based on the Discrete Logarithm Problem

In the previous chapter we learned about the RSA public key scheme As we have seen RSA is based on the hardness of factoring large integers The integer factoriza tion problem is said to be the one way function of RSA As we saw earlier roughly speaking a function is one way if it is computationally easy to compute the func

tion f x y but computationally infeasible to invert the function f 1 y x The question is whether we can find other one way functions for building asymmetric

crypto schemes It turns out that most non RSA public key algorithms with practical relevance are based on another one way function the discrete logarithm problem

In this chapter you will learn

The Diffie Hellman key exchange

Cyclic groups which are important for a deeper understanding of Diffie Hellman key exchange

The discrete logarithm problem which is of fundamental importance for many practical public key algorithms

Encryption using the Elgamal scheme

The security of many cryptographic schemes relies on the computational in tractability of finding solutions to the Discrete Logarithm Problem DLP Well known examples of such schemes are the Diffie Hellman key exchange and the Elgamal encryption scheme both of which will be introduced in this chapter Also the Elgamal digital signature scheme cf Section and the digital signature algorithm cf Section are based on the DLP as are cryptosystems based on elliptic curves Section

We start with the basic Diffie Hellman protocol which is surprisingly simple and powerful The discrete logarithm problem is defined in what are called cyclic groups The concept of this algebraic structure is introduced in Section A formal definition of the DLP as well as some illustrating examples are provided followed by a brief description of attack algorithms for the DLP With this knowledge we will revisit the Diffie Hellman protocol and more formally talk about its security We will then develop a method for encrypting data using the DLP that is known as the Elgamal cryptosystem

C Paar J Pelzl Understanding Cryptography

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

Diffie Hellman Key Exchange

The Diffie Hellman key exchange DHKE proposed by Whitfield Diffie and Mar tin Hellman in 58 was the first asymmetric scheme published in the open literature The two inventors were also influenced by the work of Ralph Merkle It provides a practical solution to the key distribution problem i e it enables two parties to derive a common secret key by communicating over an insecure chan nel1 The DHKE is a very impressive application of the discrete logarithm problem that we ll study in the subsequent sections This fundamental key agreement tech nique is implemented in many open and commercial cryptographic protocols like Secure Shell SSH Transport Layer Security TLS and Internet Protocol Security

IPSec The basic idea behind the DHKE is that exponentiation in Z p p prime is a

one way function and that exponentiation is commutative i e

k x y y x mod p

The value k x y y x mod p is the joint secret which can be used as the session key between the two parties

Let us now consider how the Diffie Hellman key exchange protocol over Z p

works In this protocol we have two parties Alice and Bob who would like to

establish a shared secret key There is possibly a trusted third party that properly chooses the public parameters which are needed for the key exchange However it is also possible that Alice or Bob generate the public parameters Strictly speaking the DHKE consists of two protocols the set up protocol and the main protocol which performs the actual key exchange The set up protocol consists of the following steps

These two values are sometimes referred to as domain parameters If Alice and Bob both know the public parameters p and computed in the set up phase they can generate a joint secret key k with the following key exchange protocol

1 The channel needs to be authenticated but that will be discussed later in this book

Here is the proof that this surprisingly simple protocol is correct i e that Alice and Bob in fact compute the same session key kAB

Proof Alice computes

Ba b a ab mod p

Ab a b ab mod p

and thus Alice and Bob both share the session key kAB ab mod p The key can now be used to establish a secure communication between Alice and Bob e g by

using kAB as key for a symmetric algorithm like AES or 3DES un

We ll look now at a simple example with small numbers

Example The Diffie Hellman domain parameters are p 29 and 2 The protocol proceeds as follows

choose a kpr A 5 choose b kpr B 12

A kpub A mod 29 B kpub B 2 mod 29

kAB Ba 75 16 mod 29 kAB Ab 16 mod 29

As one can see both parties compute the value kAB 16 which can be used as a joint secret e g as a session key for symmetric encryption

The computational aspects of the DHKE are quite similar to those of RSA Dur ing the set up phase we generate p using the probabilistic prime finding algorithms discussed in Section As shown in p should have a similar length as the RSA modulus n i e or beyond in order to provide strong security The integer needs to have a special property It should be a primitive element a topic which we discuss in the following sections The session key kAB that is being com puted in the protocol has the same bit length as p If we want to use it as a symmetric key for algorithms such as AES we can simply take the most significant bits Alternatively a hash function is sometimes applied to kAB and the output is then used as a symmetric key

During the actual protocol we first have to choose the private keys a and b They should stem from a true random generator in order to prevent an attacker from guessing them For computing the public keys A and B as well as for computing the session key both parties can make use of the square and multiply algorithm The public keys are typically precomputed The main computation that needs to be done for a key exchange is thus the exponentiation for the session key In general since the bit lengths and the computations of RSA and the DHKE are very similar they require a similar effort However the trick of using short public exponents that was shown in Section is not applicable to the DHKE

What we showed so far is the classic Diffie Hellman key exchange protocol in the group Z p where p is a prime The protocol can be generalized in particular to groups of elliptic curves This gives rise to elliptic curve cryptography which has

become a very popular asymmetric scheme in practice In order to better understand elliptic curves and schemes such as Elgamal encryption which are also closely re lated to the DHKE we introduce the discrete logarithm problem in the following sections This problem is the mathematical basis for the DHKE After we have in troduced the discrete logarithm problem we will revisit the DHKE and discuss its security

This section introduces some fundamentals of abstract algebra in particular the no tion of groups subgroups finite groups and cyclic groups which are essential for understanding discrete logarithm public key algorithms

For convenience we restate here the definition of groups which was introduced in the Chapter 4

Note that in cryptography we use both multiplicative groups i e the operation denotes multiplication and additive groups where denotes addition The latter notation is used for elliptic curves as we ll see later

Example To illustrate the definition of groups we consider the following exam ples

Z is a group i e the set of integers Z 2 1 0 1 2 together with the usual addition forms an abelian group where e 0 is the identity ele ment and a is the inverse of an element a Z

Z without 0 is not a group i e the set of integers Z without the element

0 and the usual multiplication does not form a group since there exists no inverse

a 1 for an element a Z with the exception of the elements 1 and 1

C is a group i e the set of complex numbers u iv with u v R and i2 1

together with the complex multiplication defined by

u1 iv1 u2 iv2 u v i u v

forms an abelian group The identity element of this group is e 1 and the inverse a 1 of an element a u iv C is given by a 1 u i u2 v2

However all of these groups do not play a significant role in cryptography be

cause we need groups with a finite number of elements Let us now consider the group Z n which is very important for many cryptographic schemes such as DHKE Elgamal encryption digital signature algorithm and many others

Let us verify the validity of the theorem by considering the following example

Example If we choose n 9 Z n consists of the elements 1 2 4 5 7 8

Multiplication table for Z 9

By computing the multiplication table for Z 9 depicted in Table we can eas ily check most conditions from Definition 1 Condition 1 closure is satisfied since the table only consists of integers which are elements of Z 9 For this group Conditions 3 identity and 4 inverse also hold since each row and each column of the table is a permutation of the elements of Z 9 From the symmetry along the main diagonal i e the element at row i and column j equals the element at row j

and column i we can see that Condition 5 commutativity is satisfied Condition 2 associativity cannot be directly derived from the shape of the table but follows immediately from the associativity of the usual multiplication in Zn

Finally the reader should remember from Section 1 that the inverse a 1 of each element a Z n can be computed by using the extended Euclidean algorithm

In cryptography we are almost always concerned with finite structures For instance for AES we needed a finite field We provide now the straightforward definition of a finite group

Example Examples of finite groups are

Zn the cardinality of Zn is Zn n since Zn 0 1 2 n 1

Z n remember that Z n is defined as the set of positive integers smaller than n which are relatively prime to n Thus the cardinality of Z n equals Euler s phi function evaluated for n i e Z n n For instance the group Z 9 has a car dinality of 9 32 31 6 This can be verified by the earlier example where we saw that the group consist of the six elements 1 2 4 5 7 8

The remainder of this section deals with a special type of groups namely cyclic groups which are the basis for discrete logarithm based cryptosystems We start with the following definition

We ll examine this definition by looking at an example

Example We try to determine the order of a 3 in the group Z 11 For this we keep computing powers of a until we obtain the identity element 1

a2 a a 3 3 9

a3 a2 a 9 3 27 5 mod 11

a4 a3 a 5 3 15 4 mod 11

a5 a4 a 4 3 12 1 mod 11

From the last line it follows that ord 3 5

It is very interesting to look at what happens if we keep multiplying the result by

a6 a5 a 1 a 3 mod 11 a7 a5 a2 1 a2 9 mod 11 a8 a5 a3 1 a3 5 mod 11 a9 a5 a4 1 a4 4 mod 11 a10 a5 a5 1 1 1 mod 11 a11 a10 a 1 a 3 mod 11

We see that from this point on the powers of a run through the sequence 3 9 5 4 1 indefinitely This cyclic behavior gives rise to following definition

An element of a group G with maximum order is called a generator since every element a of G can be written as a power i a of this element for some i i e generates the entire group Let us verify these properties by considering the following example

Example We want to check whether a 2 happens to be a primitive element of Z 11 1 2 3 4 5 6 7 8 9 10 Note that the cardinality of the group is Z 11 10 Let s look at all the elements that are generated by powers of the element a 2

a4 5 mod 11 a9 6 mod 11

a5 10 mod 11 a10 1 mod 11

From the last result it follows that

This implies that i a 2 is a primitive element and ii Z 11 is cyclic

We now want to verify whether the powers of a 2 actually generate all elements of the group Z 11 Let s look again at all the elements that are generated by powers of two

By looking at the bottom row we see that that the powers 2i in fact generate all elements of the group Z 11 We note that the order in which they are generated looks quite arbitrary This seemingly random relationship between the exponent i and the

group elements is the basis for cryptosystems such as the Diffie Hellman key ex change

From this example we see that the group Z 11 has the element 2 as a generator It is important to stress that the number 2 is not necessarily a generator in other cyclic groups Z n For instance in Z 7 ord 2 3 and the element 2 is thus not a generator in that group

Cyclic groups have interesting properties The most important ones for crypto graphic applications are given in the following theorems

This theorem states that the multiplicative group of every prime field is cyclic This has far reaching consequences in cryptography where these groups are the most popular ones for building discrete logarithm cryptosystems In order to under line the practical relevance of these somewhat esoteric looking theorem consider

that almost every Web browser has a cryptosystem over Z p built in

The first property is a generalization of Fermat s Little Theorem for all cyclic groups The second property is very useful in practice It says that in a cyclic group only element orders which divide the group cardinality exist

Example We consider again the group Z 11 which has a cardinality of Z 11 10 The only element orders in this group are 1 2 5 and 10 since these are the only integers that divide 10 We verify this property by looking at the order of all elements

ord 1 1 ord 6 10

ord 2 10 ord 7 10

ord 3 5 ord 8 10

ord 5 5 ord 10 2 Indeed only orders that divide 10 occur

The first property can be verified by the example above Since 10 4 the number of primitive elements is four which are the elements 2 6 7 and 8 The second property follows from the previous theorem If the group cardinality is prime the only possible element orders are 1 and the cardinality itself Since only the element 1 can have an order of one all other elements have order p

In this section we consider subsets of cyclic groups which are groups themselves Such sets are referred to as subgroups In order to check whether a subset H of a group G is a subgroup one can verify if all the properties of our group definition in Section also hold for H In the case of cyclic groups there is an easy way to generate subgroups which follows from this theorem

This theorem tells us that any element of a cyclic group is the generator of a sub group which in turn is also cyclic

Example Let us verify the above theorem by considering a subgroup of G Z 11 In an earlier example we saw that ord 3 5 and the powers of 3 generate the subset H 1 3 4 5 9 according to Theorem We verify now whether this

set is actually a group by having a look at its multiplication table

Multiplication table for the subgroup H 1 3 4 5 9

H is closed under multiplication modulo 11 Condition 1 since the table only consists of integers which are elements of H The group operation is obviously as

sociative and commutative since it follows regular multiplication rules Conditions 2 and 5 respectively The neutral element is 1 Condition 3 and for every element

a H there exists an inverse a 1 H which is also an element of H Condition 4 This can be seen from the fact that every row and every column of the table contains the identity element Thus H is a subgroup of Z 11 depicted in Figure 1

Subgroup H of the cyclic group G Z 11

More precisely it is a subgroup of prime order 5 It should also be noted that 3 is not the only generator of H but also 4 5 and 9 which follows from Theorem

An important special case are subgroups of prime order If this group cardinality is denoted by q all non one elements have order q according to Theorem

From the Cyclic Subgroup Theorem we know that each element a G of a group G generates some subgroup H By using Theorem 3 the following theorem follows

Let us now consider an application of Lagrange s theorem

Example The cyclic group Z 11 has cardinality Z 11 10 5 Thus it follows that the subgroups of Z 11 have cardinalities 1 2 5 and 10 since these are all possible divisors of 10 All subgroups H of Z 11 and their generators are given below

subgroup elements primitive elements

H3 1 3 4 5 9 3 4 5 9

The following final theorem of this section fully characterizes the subgroups of

a finite cyclic group

This theorem gives us immediately a construction method for a subgroup from a given cyclic group The only thing we need is a primitive element and the group cardinality n One can now simple compute n k and obtains a generator of the subgroup with k elements

Example 0 We again consider the cyclic group Z 11 We saw earlier that 8 is a primitive element in the group If we want to have a generator for the subgroup

of order 2 we compute

n k 8 85 8 10 mod 11

We can now verify that the element 10 in fact generates the subgroup with two elements 1 10 2 1 mod 11 3 10 mod 11 etc

Remark Of course there are smarter ways of computing 85 mod 11 e g through 85 2 32 10 mod 11

The Discrete Logarithm Problem

After the somewhat lengthy introduction to cyclic groups one might wonder how they are related to the rather straightforward DHKE protocol It turns out that the underlying one way function of the DHKE the discrete logarithm problem DLP can directly be explained using cyclic groups

The Discrete Logarithm Problem in Prime Fields

We start with the DLP over Z p where p is a prime

Remember from Section that such an integer x must exist since is a primi tive element and each group element can be expressed as a power of any primitive element This integer x is called the discrete logarithm of to the base and we can formally write

Computing discrete logarithms modulo a prime is a very hard problem if the param eters are sufficiently large Since exponentiation x mod p is computationally easy this forms a one way function

Example 1 We consider a discrete logarithm in the group Z 47 in which 5 is a primitive element For 41 the discrete logarithm problem is Find the positive

Even for such small numbers determining x is not entirely straightforward By using a brute force attack i e systematically trying all possible values for x we obtain the solution x 15

In practice it is often desirable to have a DLP in groups with prime cardinality in order to prevent the Pohlig Hellman attack cf Section Since groups Z p have cardinality p 1 which is obviously not prime one often uses DLPs in subgroups of Z p with prime order rather than using the group Z p itself We illustrate this with an example

Example 2 We consider the group Z 47 which has order 46 The subgroups in Z 47 have thus a cardinality of 23 2 and 1 2 is an element in the subgroup with 23 elements and since 23 is a prime is a primitive element in the subgroup

A possible discrete logarithm problem is given for 36 which is also in the subgroup Find the positive integer x 1 x 23 such that

By using a brute force attack we obtain a solution for x 17

The Generalized Discrete Logarithm Problem

The feature that makes the DLP particularly useful in cryptography is that it is not restricted to the multiplicative group Z p p prime but can be defined over any cyclic groups This is called the generalized discrete logarithm problem GDLP and can

be stated as follows

As in the case of the DLP in Z p such an integer x must exist since is a primi

tive element and thus each element of the group G can be generated by repeated application of the group operation on

It is important to realize that there are cyclic groups in which the DLP is not difficult Such groups cannot be used for a public key cryptosystem since the DLP is not a one way function Consider the following example

Example 3 This time we consider the additive group of integers modulo a prime For instance if we choose the prime p 11 G Z11 is a finite cyclic group with the primitive element 2 Here is how generates the group

We try now to solve the DLP for the element 3 i e we have to compute the integer 1 x 11 such that

x 2 2 2 2 3 mod 11

Here is how an attack against this DLP works Even though the group operation is addition we can express the relationship between and the discrete logarithm x in terms of multiplication

In order to solve for x we simply have to invert the primitive element

Using e g the extended Euclidean algorithm we can compute 6 mod 11 from which the discrete logarithm follows as

The discrete logarithm can be verified by looking at the small table provided above We can generalize the above trick to any group Zn for arbitrary n and ele ments Zn Hence we conclude that the generalized DLP is computationally easy over Zn The reason why the DLP can be solved here easily is that we have mathematical operations which are not in the additive group namely multiplication

After this counterexample we now list discrete logarithm problems that have been proposed for use in cryptography

The multiplicative group of the prime field Zp or a subgroup of it For instance the classical DHKE uses this group but also Elgamal encryption or the Digital Signature Algorithm DSA These are the oldest and most widely used types of discrete logarithm systems

The cyclic group formed by an elliptic curve Elliptic curve cryptosystems are introduced in Chapter 9 They have become popular in practice over the last decade

The multiplicative group of a Galois field GF 2m or a subgroup of it These

groups can be used completely analogous to multiplicative groups of prime fields and schemes such as the DHKE can be realized with them They are not as pop ular in practice because the attacks against them are somewhat more powerful than those against the DLP in Zp Hence DLPs over GF 2m require somewhat higher bit lengths for providing the same level of security than those over Zp

Hyperelliptic curves or algebraic varieties which can be viewed as generalization as elliptic curves They are currently rarely used in practice but in particular hyperelliptic curves have some advantages such as short operand lengths

There have been proposals for other DLP based cryptosystems over the years but none of them have really been of interest in practice Often it was found that the underlying DL problem was not difficult enough

Attacks Against the Discrete Logarithm Problem

This section introduce methods for solving discrete logarithm problems Readers only interested in the constructive use of DL schemes can skip this section

As we have seen the security of many asymmetric primitives is based on the difficulty of computing the DLP in cyclic groups i e to compute x for a given and in G such that

x

holds We still do not know the exact difficulty of computing the discrete logarithm x in any given actual group What we mean by this is that even though some at tacks are known one does not know whether there are any better more powerful algorithms for solving the DLP This situation is similar to the hardness of integer factorization which is the one way function underlying RSA Nobody really knows what the best possible factorization method is For the DLP some interesting gen eral results exist regarding its computational hardness This section gives a brief overview of algorithms for computing discrete logarithms which can be classified into generic algorithms and nongeneric algorithms and which will be discussed in a little more detail

Generic DL algorithms are methods which only use the group operation and no other algebraic structure of the group under consideration Since they do not exploit special properties of the group they work in any cyclic group Generic algorithms for the discrete logarithm problem can be subdivided into two classes The first class encompasses algorithms whose running time depends on the size of the cyclic group like the brute force search the baby step giant step algorithm and Pollard s rho method The second class are algorithms whose running time depends on the size of the prime factors of the group order like the Pohlig Hellman algorithm

A brute force search is the most na ve and computationally costly way for comput ing the discrete logarithm log We simply compute powers of the generator successively until the result equals

For a random logarithm x we do expect to find the correct solution after checking half of all possible x This gives us a complexity of O G steps2 where G is the cardinality of the group

To avoid brute force attacks on DL based cryptosystems in practice the cardi nality G of the underlying group must thus be sufficiently large For instance in the case of the group Z p p prime which is the basis for the DHKE p 1 2 tests

are required on average to compute a discrete logarithm Thus G p 1 should

be at least in the order of to make a brute force search infeasible using today s computer technology Of course this consideration only holds if a brute force attack is the only feasible attack which is never the case There exist much more powerful algorithms to solve discrete logarithms as we will see below

Shanks Baby Step Giant Step Method

Shanks algorithm is a time memory tradeoff method which reduces the time of a brute force search at the cost of extra storage The idea is based on rewriting the discrete logarithm x log in a two digit representation

x xg m xb for 0 xg xb m The value m is chosen to be of the size of the square root of the group order i e

m G We can now write the discrete logarithm as x xg m xb which leads to

m xg xb

The idea of the algorithm is to find a solution xg xb for Eq from which the discrete logarithm then follows directly according to Eq The core idea for the algorithm is that Eq can be solved by searching for xg and xb separatedly i e

using a divide and conquer approach In the first phase of the algorithm we compute and s tore all values xb where 0 xb m This is the baby step phase that requires

In the giant step phase the algorithm checks for all xg in the range 0 xg m

whether the following condition is fulfilled

for some stored entry xb that was computed during the baby step phase In case of a match i e m xg 0 xb 0 for some pair xg 0 xb 0 the discrete logarithm is given by

The baby step giant step method requires O G computational steps and an equal amount of memory In a group of order an attacker would only need

2 We use the popular big Oh notation here A complexity function f x has big Oh notation

O g x if f x c g x for some constant c and for input values x greater than some value x0

approximately computations and memory locations which is easily

achievable with today s PCs and hard disks Thus in order to obtain an attack com plexity of a group must have a cardinality of at least G In the case of

groups G Z p the prime p should thus have at least a length of bit However as we see below there are more powerful attacks against DLPs in Z p which forces

even larger bit lengths of p

Pollard s rho method has the same expected run time O G as the baby step giant step algorithm but only negligible space requirements The method is a prob abilistic algorithm which is based on the birthday paradox cf Section 3 We will only sketch the algorithm here The basic idea is to pseudorandomly generate group elements of the form i j For every element we keep track of the values i and j We continue until we obtain a collision of two elements i e until we have

i1 j1 i2 j2

If we substitute x and compare the exponents on both sides of the equation the collision leads to the relation i1 x j1 i2 x j2 mod G Note that we are in a cyclic group with G elements and have to take the exponent modulo G From here the discrete logarithm can easily computed as

x i2 i1 mod G j1 j2

An important detail which we omit here is the exact way to find the collision In any case the pseudorandom generation of the elements is a random walk through the group This can be illustrated by the shape of the Greek letter rho hence the name of this attack

Pollard s rho method is of great practical importance because it is currently the best known algorithm for computing discrete logarithms in elliptic curve groups Since the method has an attack complexity of O G computations elliptic curve groups should have a size of at least In fact elliptic curve cryptosystems with bit operands are very popular in practice

There are still much more powerful attacks known for the DLP in Z p as we will

Pohlig Hellman Algorithm

The Pohlig Hellman method is an algorithm which is based on the Chinese Re mainder Theorem not introduced in this book it exploits a possible factorization of the order of a group It is typically not used by itself but in conjunction with any of the other DLP attack algorithms in this section Let

G p1 p2 pl

be the prime factorization of the group order G Again we attempt to compute a discrete logarithm x log in G This is also a divide and conquer algorithm The basic idea is that rather than dealing with the large group G one computes

smaller discrete logarithms xi x mod pei in the subgroups of order pei The desired

discrete logarithm x can then be computed from all xi i 1 l by using the Chinese Remainder Theorem Each individual small DLP xi can be computed using Pollard s rho method or the baby step giant step algorithm

The run time of the algorithm clearly depends on the prime factors of the group order To prevent the attack the group order must have its largest prime factor in the range of An important practical consequence of the Pohlig Hellman algorithm is that one needs to know the prime factorization of the group order Especially in the case of elliptic curve cryptosystems computing the order of the cyclic group is not always easy

Nongeneric Algorithms The Index Calculus Method

All algorithms introduced so far are completely independent of the group being attacked i e they work for discrete logarithms defined over any cyclic group Non generic algorithms efficiently exploit special properties i e the inherent structure of certain groups This can lead to much more powerful DL algorithms The most important nongeneric algorithm is the index calculus method

Both the baby step giant step algorithm and Pollard s rho method have a run time which is exponential in the bit length of the group order namely of about 2n 2 steps where n is the bit length of G This greatly favors the crypto designer over the cryptanalyst For instance increasing the group order by a mere 20 bit increases the attack effort by a factor of This is a major reason why elliptic curves have better long term security behavior than RSA or cryptosystems based on the

DLP in Z p The question is whether there are more powerful algorithms for DLPs

in certain specific groups The answer is yes

The index calculus method is a very efficient algorithm for computing discrete logarithms in the cyclic groups Z p and GF 2m It has a subexponential running time We will not introduce the method here but just provide a very brief description

The index calculus method depends on the property that a significant fraction of elements of G can be efficiently expressed as products of elements of a small subset

of G For the group Z p this means that many elements should be expressable as a product of small primes This property is satisfied by the groups Z p and GF 2m

However one has not found a way to do the same for elliptic curve groups The

index calculus is so powerful that in order to provide a security of 80 bit i e an attacker has to perform steps the prime p of a DLP in Z p should be at least bit long Table gives an overview on the DLP records achieved since the

early s The index calculus method is somewhat more powerful for solving the DLP in GF 2m Hence the bit lengths have to be chosen somewhat longer to

achieve the same level of security For that reason DLP schems in GF 2m are not as widely used in practice

Summary of records for computing discrete logarithms in Z p

Security of the Diffie Hellman Key Exchange

Security of the Diffie Hellman Key Exchange

After the introduction of the discrete logarithm problem we are now well prepared to discuss the security of the DHKE from Section First it should be noted that a protocol that uses the basic version of the DHKE is not secure against active attacks This means if an attacker Oscar can either modify messages or generate false mes sages Oscar can defeat the protocol This is called man in the middle attack and is described in Section

Let s now consider the possibilities of a passive adversary i e Oscar can only listen but not alter messages His goal is to compute the session key kAB shared by Alice and Bob Which information does Oscar get from observing the proto col Certainly Oscar knows and p because these are public parameters chosen during the set up protocol In addition Oscar can obtain the values A kpub A and B kpub B by eavesdropping on the channel during an execution of the key exchange protocol Thus the question is whether he is capable of computing k ab from

p A a mod p and B b mod p This problem is called the Diffie Hellman problem DHP Like the discrete logarithm problem it can be generalized to arbi trary finite cyclic groups Here is a more formal statement of the DHP

One general approach to solving the Diffie Hellman problem is as follows For il lustration purposes we consider the DHP in the multiplicative group Z p Suppose and that s a big suppose Oscar knows an efficient method for computing discrete logarithms in Z p Then he could also solve the Diffie Hellman problem and obtain the key kAB via the following two steps

Compute Alice s private key a kpr A by solving the discrete logarithm problem

Compute the session key kAB B

But as we know from Section even though this looks easy computing the discrete logarithm problem is infeasible if p is sufficiently large

At this point it is important to note that it is not known whether solving the DLP is the only way to solve the DHP In theory it is possible that there exists another method for solving the DHP without computing the discrete logarithm We note that the situation is analogous to RSA where it is also not known whether factoring is the best way of breaking RSA However even though it is not proven in a mathematical sense it is often assumed that solving the DLP efficiently is the only way for solving the DHP efficiently

Hence in order to assure the security of the DHKE in practice we have to ascer tain that the corresponding DLP cannot be solved This is achieved by choosing p

large enough so that the index calculus method cannot compute the DLP By con sulting we see that a security level of 80 bit is achieved by primes of lengths bit and for bit security we need about bit An additional requirement is that in order to prevent the Pohlig Hellman attack the order p 1 of the cyclic group must not factor in only small prime factors Each of the subgroups formed by the factors of p 1 can be attacked using the baby step giant step method or Pollards s rho method but not by the index calculus method Hence the smallest prime factor of p 1 must be at least bit long for an 80 bit security level and at least bit long for a security level of bit

The Elgamal Encryption Scheme

The Elgamal encryption scheme was proposed by Taher Elgamal in 73 It is also often referred to as Elgamal encryption It can be viewed as an extension of the DHKE protocol Not surprisingly its security is also based on the intractability of the discrete logarithm problem and the Diffie Hellman problem We consider the

Elgamal encryption scheme over the group Z p where p is a prime However it can

be applied to other cyclic groups too in which the DL and DH problem is intractable for instance in the multiplicative group of a Galois field GF 2m

From Diffie Hellman Key Exhange to Elgamal Encryption

In order to understand the Elgamal scheme it is very helpful to see how it follows almost immediately from the DHKE We consider two parties Alice and Bob If Alice wants to send an encrypted message x to Bob both parties first perform a Diffie Hellman key exchange to derive a shared key kM For this we assume that a large prime p and a primitive element have been generated Now the new idea is that Alice uses this key as a multiplicative mask to encrypt x as y x kM mod p This process is depicted below

The protocol consists of two phases the classical DHKE Steps a f which is followed by the message encryption and decryption Steps g and h respectively Bob computes his private key d and public key This key pair does not change i e it can be used for encrypting many messages Alice however has to generate a new public private key pair for the encryption of every message Her private key is denoted by i and her public key by kE The latter is an ephemeral existing only temporarily key hence the index E The joint key is denoted by kM because it is used for masking the plaintext

For the actual encryption Alice simply multiplies the plaintext message x by the masking key kM in Z p On the receiving side Bob reverses the encryption by multipliying with the inverse mask Note that one property of cyclic groups is that given any key kM Z p every messages x maps to another ciphertext if the two values are multiplied Moreover if the key kM is randomly drawn from Z p every ciphertext y 1 2 p 1 is equally likely

We provide now a somewhat more formal description of the scheme We distinguish three phases The set up phase is executed once by the party who issues the public key and who will receive the message The encryption phase and the decryption phase are executed every time a message is being sent In contrast to the DHKE no trusted third party is needed to choose a prime and primitive element Bob generates them and makes them public by placing them in a database or on his website

The actual Elgamal encryption protocol rearranges the sequence of operations from the na ve Diffie Hellman inspired approach we saw before The reason for this is that Alice has to send only one message to Bob as opposed to two messages in the earlier protocol

The ciphertext consists of two parts the ephemeral key kE and the masked plain text y Since in general all parameters have a bit length of log2 p the ciphertext kE y is twice as long as the message Thus the message expansion factor of Elga mal encryption is two

We prove now the correctness of the Elgamal protocol

Proof We have to show that dkpr kE y actually yields the original message x

dkpr kE y y kM 1 mod p

x kM kd 1 mod p

x d i i d 1 mod p

x d i d i x mod p

Let s look at an example with small numbers

Example 4 In this example Bob generates the Elgamal keys and Alice encrypts the message x 26

message x 26 generate p 29 and 2

choose kpr B d 12

compute d 7 mod 29

compute kE i 3 mod 29 compute kM i 16 mod 29 encrypt y x kM 10 mod 29

compute kM kd 16 mod 29

x y k 1 10 20 26 mod 29

It is important to note that unlike the schoolbook version of the RSA scheme

Elgamal is a probabilistic encryption scheme i e encrypting two identical mes sages x1 and x2 where x1 x2 Z p using the same public key results with extremely high likelihood in two different ciphertexts y1 y2 This is because i is chosen at random from 2 3 p 2 for each encryption and thus also the session key kM i used for encryption is chosen at random for each encryption In this way a brute force search for x is avoided a priori

Computational Aspects

Key Generation During the key generation by the receiver Bob in our example a prime p must be generated and the public and private have to be computed Since the security of Elgamal also depends on the discrete logarithm problem p needs to have the properties discussed in Section In particular it should have a length of at least bits To generate such a prime the prime finding algorithms discussed in Section can be used The private key should be generated by a true random number generater The public key requires one exponentiation for which the square and multiply algorithm is used

Encryption Within the encryption procedure two modular exponentiations and one modular multiplication are required for computing the ephemeral and the masking key as well as for the message encryption All operands involved have a bit length of log2 p For efficient exponentiation one should apply the square and multiply algorithm that was introduced in Section It is important to note that the two ex ponentiations which constitute almost all computations necessary are independent of the plaintext Hence in some applications they can be precomputed at times of low computational load stored and used when the actual encryption is needed This can be a major advantage in practice

Decryption The main steps of the decryption are first an exponentiation kM

kd mod p using the square and multiply algorithm followed by an inversion of kM

that is performed with the extended Euclidean algorithm However there is a short cut based on Fermat s Little Theorem that combines these two steps in a single one

From the theorem which was introduced in Section 4 follows that

for all kE Z p We can now merge Step 1 and 2 of the decryption as follows

The equivalence relation allows us to compute the inverse of the masking key using a single exponentiation with the exponent p d 1 After that one mod

ular multiplication is required to recover x y k 1 mod p As a consequence de cryption essentially requires one execution of the square and multiply algorithm

followed by a single modular multiplication for recovering the plaintext

If we want to assess the security of the Elgamal encryption scheme it is important to distinguish between passive i e listen only and active attacks which allow Oscar to generate and alter messages

The security of the Elgamal encryption scheme against passive attacks i e recover ing x from the information p d kE i and y x i obtained by eaves dropping relies on the hardness of the Diffie Hellman problem cf Section 4 Currently there is no other method known for solving the DHP than computing discrete logarithms If we assume Oscar has supernatural powers and can in fact compute DLPs he would have two ways of attacking the Elgamal scheme

Recover x by finding Bob s secret key d

This step solves the DLP which is computationally infeasible if the parame ters are chosen correctly However if Oscar succeeds with it he can decrypt the plaintext by performing the same steps as the receiver Bob

x y kd 1 mod p

Alternatively instead of computing Bob s secret exponent d Oscar could attempt to recover Alice s random exponent i

Again this step is solving the discrete logarithm problem Should Oscar succeed with it he can compute the plaintext as

x y i 1 mod p

In both cases Oscar has to solve the DL problem in the finite cyclic group Z p In contrast to elliptic curves the more powerful index calculus method Section can be applied here Thus in order to guarantee the security of the Elgamal scheme over Z p today p should at least have a length of bits

Just as in the DHKE protocol we have to be careful that we do not fall vicitim to

what is a called a small subgroup attack In order to counter this attack in practice primitive elements are used which generate a subgroup of prime order In such groups all elements are primitive and small subgroups do not exist One of the problems illustrates the pitfalls of a small subgroup attack with an example

Like in every other asymmetric scheme it must be assured that the public keys are authentic This means that the encrypting party Alice in our example in fact has the public key that belongs to Bob If Oscar manages to convince Alice that his key is Bob s he can easily attack the scheme In order to prevent the attack certificates can be used a topic that is discussed in Chapter 13

Another weakness if not necessarily an attack that requires any direct action by Oscar of the Elgamal encryption is that the secret exponent i should not be reused Assume Alice used the value i for encrypting two subsequent messages x1 and x2 In this case the two masking keys would be the same namely kM i Also the two ephemeral keys would be identical She would send the two ciphertexts y1 kE

and y1 kE over the channel If Oscar knows or can guess the first message he can compute the masking key as kM y 1 mod p With this he can decrypt x2

Any other message encrypted with the same i value can also be recovered this way As a consequence of this attack one has to take care that the secret exponent i does not repeat For instance if one would use a cryptographically secure PRNG as introduced in Section 1 but with the same seed value every time a session is initiated the same sequence of i values would be used for every encryption a fact that could be exploited by Oscar Note that Oscar can detect the re use of secret exponents because they lead to identical ephemeral keys

Another active attack against Elgamal exploits its malleability If Oscar observes the ciphertext kE y he can replace it by

where s is some integer The receiver would compute

dkpr kE s y sy k 1 mod p

s x kM k 1 mod p

Thus the decrypted text is also a multiple of s The situation is exactly the same as for the attack that exploited the malleability of RSA which was introduced in Section Oscar is not able to decrypt the ciphertext but he can manipulated it in a specific way For instance he could double or triple the integer value of the decryption result by choosing s equal to 2 or 3 respectively As it was the case for RSA schoolbook Elgamal encryption is often not used in practice but some padding is introduced to prevent these types of attacks

Discussion and Further Reading

Diffie Hellman Key Exchange and Elgamal Encryption The DHKE was intro duced in the landmark paper 58 which also described the concept of public key cryptography Due to the independent discovery of asymmetric cryptography by Ralph Merkle Hellman suggested in that the algorithm should be named Diffie Hellman Merkle key exchange In Chapter 13 of this book a more de tailed treatment of key exchanges based on the DHKE is provided The scheme is standardized in ANSI X 2 5 and is used in numerous security protocols such as TLS One of the attractive features of DHKE is that it can be generalized to any cyclic group not only to the multiplicative group of a prime field that was shown in this chapter In practice the most popular group in addition to Z is the DHKE over

an elliptic curve that is presented in Section

The DHKE is a two party protocol It can be extended to a group key agreement in which more than two parties establish a joint Diffie Hellman key see e g 38 The Elgamal encryption as proposed in by Tahar Elgamal 73 is widely used For example Elgamal is part of the free GNU Privacy Guard GnuPG OpenSSL Pretty Good Privacy PGP and other crypto software Active attacks against the Elgamal encryption scheme such as those discussed in Section have quite strong requirements that have to be fulfilled which is quite difficult in reality There exist schemes which are related to Elgamal but have stronger security properties These include e g the Cramer Shoup System 49 and the DHAES 1 scheme proposed by Abdalla Bellare and Rogaway these are secure against chosen

ciphertext attacks under certain assumptions

Discrete Logarithm Problem This chapter sketched the most important attack al gorithms for solving discrete logarithm problems A good overview on these in cluding further references are given in p ff We also discussed the re lationship between the Diffie Hellman problem DHP and the discrete logarithm problem DLP This relationship is a matter of great importance for the foundations of cryptography Key contributions which study this are 31

The idea of using the DLP in groups other than Z is exploited in elliptic curve

cryptography a topic that is treated in Chapter 9 Other cryptoystems based on the generalized DLP include hyperelliptic curves a comprehensive treatment of which can be found in 44 Rather than using the prime field Z it is also possible to use certain extension fields which offer computational advantages Two of the better studied DL systems over extension fields are Lucas Based Cryptosystems 26 and Efficient and Compact Subgroup Trace Representation XTR

The Diffie Hellman protocol is a widely used method for key exchange It is based on cyclic groups

The discrete logarithm problem is one of the most important one way functions in modern asymmetric cryptography Many public key algorithms are based on it

In practice the multiplicative group of the prime field Zp or the group of an elliptic curve are used most often

For the Diffie Hellman protocol in Z p the prime p should be at least bits

long This provides a security roughly equivalent to an 80 bit symmetric cipher

For a better long term security a prime of length bits should be chosen

The Elgamal scheme is an extension of the DHKE where the derived session key is used as a multiplicative masked to encrypt a message

Elgamal is a probabilistic encryption scheme i e encrypting two identical mes sages does not yield two identical ciphertexts

For the Elgamal encryption scheme over Z p the prime p should be at least

bits long i e p 0

Understanding the functionality of groups cyclic groups and subgroups is im portant for the use of public key cryptosystems based on the discrete logarithm problem That s why we are going to practice some arithmetic in such structures in this set of problems

Let s start with an easy one Determine the order of all elements of the multi plicative groups of

Create a list with two columns for every group where each row contains an element

a and the order ord a

Hint In order to get familiar with cyclic groups and their properties it is a good idea to compute all orders by hand i e use only a pocket calculator If you want to refresh your mental arithmetic skills try not to use a calculator whenever possible in particular for the first two groups

We consider the group Z 53 What are the possible element orders How many elements exist for each order

We now study the groups from Problem

How many elements does each of the multiplicative groups have

Do all orders from above divide the number of elements in the corresponding multiplicative group

Which of the elements from Problem are primitive elements

Verify for the groups that the number of primitive elements is given by Z p

In this exercise we want to identify primitive elements generators of a multi plicative group since they play a big role in the DHKE and and many other public

key schemes based on the DL problem You are given a prime p and the corresponding multiplicative group Z

Determine how many generators exist in Z

What is the probability of a randomly chosen element a Z being a genera

Determine the smallest generator a Z with a

Hint The identification can be done na vely through testing all possible factors

of the group cardinality p 1 or more efficiently by checking the premise that a p 1 qi 1 mod p for all prime factors qi with p 1 qei You can simply start with a and repeat these steps until you find a respective generator of

What measures can be taken in order to simplify the search for generators for arbitrary groups Z p

Compute the two public keys and the common key for the DHKE scheme with the parameters p 2 and

In all cases perform the computation of the common key for Alice and Bob This is also a perfect check of your results

We now design another DHKE scheme with the same prime p as in Problem This time however we use the element 4 The element 4 has order and generates thus a subgroup with elements Compute kAB for

Why are the session keys identical

In the DHKE protocol the private keys are chosen from the set

Why are the values 1 and p 1 excluded Describe the weakness of these two values

Given is a DHKE algorithm The modulus p has bit and is a generator of a subgroup where ord

What is the maximum value that the private keys should have

How long does the computation of the session key take on average if one modular multiplication takes s and one modular squaring s Assume that the public keys have already been computed

One well known acceleration technique for discrete logarithm systems uses short primitive elements We assume now that is such a short element e g a 16 bit integer Assume that modular multiplication with takes now only 30 s How long does the computation of the public key take now Why is the time for one modular squaring still the same as above if we apply the square and multiply algorithm

We now want to consider the importance of the proper choice of generators in multiplicative groups

Show that the order of an element a Zp with a p 1 is always 2

What subgroup is generated by a

Briefly describe a simple attack on the DHKE which exploits this property

We consider a DHKE protocol over a Galois fields GF 2m All arithmetic is done in GF 25 with P x x5 x2 1 as an irreducible field polynomial The primitive element for the Diffie Hellman scheme is x2 The private keys are a 3 and b 12 What is the session key kAB

In this chapter we saw that the Diffie Hellman protocol is as secure as the Diffie Hellman problem which is probably as hard as the DL problem in the group Z p However this only holds for passive attacks i e if Oscar is only capable

of eavesdropping If Oscar can manipulate messages between Alice and Bob the

key agreement protocol can easily be broken Develop an active attack against the Diffie Hellman key agreement protocol with Oscar being the man in the middle

Write a program which computes the discrete logarithm in Z p by exhaustive search The input parameters for your program are p The program computes x where x mod p

Compute the solution to log1 in Z 1

Encrypt the following messages with the Elgamal scheme p and

kpr d i x 33

kpr d i x 33

kpr d i 45 x

kpr d i 47 x

Now decrypt every ciphertext and show all steps

Assume Bob sends an Elgamal encrypted message to Alice Wrongly Bob uses the same parameter i for all messages Moreover we know that each of Bob s cleartexts start with the number x1 21 Bob s ID We now obtain the following ciphertexts

The Elgamal parameters are p 31 3 18 Determine the second plaintext

Given is an Elgamal crypto system Bob tries to be especially smart and chooses the following pseudorandom generator to compute new i values

i j i j 1 f j 1 j where f j is a complicated but known pseudorandom function for instance f j could be a cryptographic hash function such as SHA or RIPE MD i0 is a true

random number that is not known to Oscar

Bob encrypts n messages x j as follows

kEj ij mod p y j x j ij mod p

where 1 j n Assume that the last cleartext xn is known to Oscar and all cipher text

Provide a formula with which Oscar can compute any of the messages x j 1 j n 1 Of course following Kerckhoffs principle Oscar knows the construction method shown above including the function f

Given an Elgamal encryption scheme with public parameters kpub p and an unknown private key kpr d Due to an erroneous implementation of the random number generator of the encrypting party the following relation holds for two temporary keys

Given n consecutive ciphertexts

kE1 y1 kE2 y2 kEn yn

Furthermore the first plaintext x1 is known e g header information

Describe how an attacker can compute the plaintexts x1 x2 xn from the given quantities

Can an attacker compute the private key d from the given information Give reasons for your answer

Considering the four examples from Problem 3 we see that the Elgamal scheme is nondeterministic A given plaintext x has many valid ciphertexts e g both x 33 and x have the same ciphertext in the problem above

Why is the Elgamal signature scheme nondeterministic

How many valid ciphertexts exist for each message x general expression How many are there for the system in Problem 3 numerical answer

Is the RSA crypto system nondeterministic once the public key has been chosen

We investigate the weaknesses that arise in Elgamal encryption if a public key of small order is used We look at the following example Assume Bob uses the group Z 29 with the primitive element 2 His public key is 28

What is the order of the public key

Which masking keys kM are possible

Alice encrypts a text message Every character is encoded according to the simple rule a 0 z 25 There are three additional ciphertext symbols a 26

o 27 u 28 She transmits the following 11 ciphertexts kE y

3 15 19 14 6 15 1 24 22 13 4 7

13 24 3 21 18 12 26 5 7 12

Decrypt the message without computing Bob s private key Just look at the ci phertext and use the fact that there are only very few masking keys and a bit of guesswork

Elliptic Curve Cryptosystems

Elliptic Curve Cryptography ECC is the newest member of the three families of established public key algorithms of practical relevance introduced in Sect 3 However ECC has been around since the mid s

ECC provides the same level of security as RSA or discrete logarithm systems with considerably shorter operands approximately 6 bit vs bit ECC is based on the generalized discrete logarithm problem and thus DL protocols such as the Diffie Hellman key exchange can also be realized using elliptic curves In many cases ECC has performance advantages fewer computations and band width advantages shorter signatures and keys over RSA and Discrete Logarithm DL schemes However RSA operations which involve short public keys as intro duced in Sect 1 are still much faster than ECC operations

The mathematics of elliptic curves are considerably more involved than those of RSA and DL schemes Some topics e g counting points on elliptic curves go far beyond the scope of this book Thus the focus of this chapter is to explain the basics of ECC in a clear fashion without too much mathematical overhead so that the reader gains an understanding of the most important functions of cryptosystems based on elliptic curves

In this chapter you will learn

The basic pros and cons of ECC vs RSA and DL schemes

What an elliptic curve is and how to compute with it

How to build a DL problem with an elliptic curve

Protocols that can be realized with elliptic curves

Current security estimations of cryptosystems based on elliptic curves

How to Compute with Elliptic Curves

We start by giving a short introduction to the mathematical concept of elliptic curves independent of their cryptographic applications ECC is based on the gener alized discrete logarithm problem Hence what we try to do first is to find a cyclic

C Paar J Pelzl Understanding Cryptography

DOI 0 6 Qc Springer Verlag Berlin Heidelberg

group on which we can build our cryptosystem Of course the mere existence of a cyclic group is not sufficient The DL problem in this group must also be computa tionally hard which means that it must have good one way properties

We start by considering certain polynomials e g functions with sums of expo nents of x and y and we plot them over the real numbers

Example Let s look at the polynomial equation x2 y2 r2 over the real num bers R If we plot all the pairs x y which fulfill this equation in a coordinate sys

Plot of all points x y which fulfill the equation x2 y2 r2 over R

tem we obtain a circle as shown in Fig

We now look at other polynomial equations over the real numbers

Example A slight generalization of the circle equation is to introduce coeffi cients to the two terms x2 and y2 i e we look at the set of solutions to the equation a x2 b y2 c over the real numbers It turns out that we obtain an ellipse as

Plot of all points x y which fulfill the equation a x2 b y2 c over R

Definition of Elliptic Curves

From the two examples above we conclude that we can form certain types of curves from polynomial equations By curves we mean the set of points x y which are

solutions of the equations For example the point x r y 0 fulfills the equation of a circle and is thus in the set The point x r 2 y r 2 is not a solution to the polynomial x2 y2 r2 and is thus not a set member An elliptic curve is a special type of polynomial equation For cryptographic use we need to consider the curve not over the real numbers but over a finite field The most popular choice is prime fields GF p cf Sect where all arithmetic is performed modulo a prime p

The definition of elliptic curve requires that the curve is nonsingular Geometri cally speaking this means that the plot has no self intersections or vertices which is achieved if the discriminant of the curve a3 is nonzero

For cryptographic use we are interested in studying the curve over a prime field as in the definition However if we plot such an elliptic curve over Zp we do not get anything remotely resembling a curve However nothing prevents us from taking an elliptic curve equation and plotting it over the set of real numbers

Example In Figure the elliptic curve y2 x x 3 is shown over the real numbers

y2 x3 3x 3 over R

We notice several things from this elliptic curve plot 1 First the elliptic curve is symmetric with respect to the x axis This follows directly from the fact that for all values xi which are on the elliptic curve both yi x3 a xi b and yli

x3 a xi b are solutions Second there is one intersection with the x axis This follows from the fact that it is a cubic equation if we solve for y 0 which has

one real solution the intersection with the x axis and two complex solutions which do not show up in the plot There are also elliptic curves with three intersections with the x axis

We now return to our original goal of finding a curve with a large cyclic group which is needed for constructing a discrete logarithm problem The first task for finding a group is done namely identifying a set of elements In the elliptic curve case the group elements are the points that fulfill Eq The next question at hand is How do we define a group operation with those points Of course we have to make sure that the group laws from Definition 1 in Sect hold for the operation

Group Operations on Elliptic Curves

Let s denote the group operation with the addition symbol2 Addition means that given two points and their coordinates say P x1 y1 and Q x2 y2 we have to compute the coordinates of a third point R such that

x1 y1 x2 y2 x3 y3

As we will see below it turns out that this addition operation looks quite arbi trary Luckily there is a nice geometric interpretation of the addition operation if we consider a curve defined over the real numbers For this geometric interpretation we have to distinguish two cases the addition of two distinct points named point addition and the addition of one point to itself named point doubling

Point Addition P Q This is the case where we compute R P Q and P

Q The construction works as follows Draw a line through P and Q and obtain a third point of intersection between the elliptic curve and the line Mirror this third intersection point along the x axis This mirrored point is by definition the point R Figure shows the point addition on an elliptic curve over the real numbers

Point Doubling P P This is the case where we compute P Q but P Q Hence we can write R P P 2P We need a slightly different construction here We

1 Note that elliptic curves are not ellipses They play a role in determining the circumference of ellipses hence the name

2 Note that the choice of naming the operation addition is completely arbitrary we could have also called it multiplication

Point addition on an elliptic curve over the real numbers

draw the tangent line through P and obtain a second point of intersection between this line and the elliptic curve We mirror the point of the second intersection along the x axis This mirrored point is the result R of the doubling Figure shows the

Point doubling on an elliptic curve over the real numbers

doubling of a point on an elliptic curve over the real numbers

You might wonder why the group operations have such an arbitrary looking form Historically this tangent and chord method was used to construct a third point if two points were already known while only using the four standard algebraic op erations add subtract multiply and divide It turns out that if points on the elliptic curve are added in this very way the set of points also fulfill most conditions neces sary for a group that is closure associativity existence of an identity element and existence of an inverse

Of course in a cryptosystem we cannot perform geometric constructions How ever by applying simple coordinate geometry we can express both of the geomet

ric constructions from above through analytic expressions i e formulae As stated above these formulae only involve the four basic algebraic operations These op erations can be performed in any field not only over the field of the real numbers cf Sect In particular we can take the curve equation from above but we now consider it over prime fields GF p rather than over the real numbers This yields the following analytical expressions for the group operation

Note that the parameter s is the slope of the line through P and Q in the case of point addition or the slope of the tangent through P in the case of point doubling

Even though we made major headway towards the establishment of a finite group we are not there yet One thing that is still missing is an identity or neutral element O such that

for all points P on the elliptic curve It turns out that there isn t any point x y that fulfills the condition Instead we define an abstract point at infinity as the neutral element O This point at infinity can be visualized as a point that is located towards plus infinity along the y axis or towards minus infinity along the y axis

According the group definition we can now also define the inverse P of any group element P as

The question is how do we find P If we apply the tangent and chord method from above it turns out that the inverse of the point P xp yp is the point P xp yp i e the point that is reflected along the x axis Figure shows the point P together with its inverse Note that finding the inverse of a point P xp yp is now trivial We simply take the negative of its y coordinate In the case of elliptic

curves over a prime field GF p the most interesting case in cryptography this is easily achieved since yp p yp mod p hence

Now that we have defined all group properties for elliptic curves we now look at an example for the group operation

Example We consider a curve over the small field Z17

The inverse of a point P on an elliptic curve

E y2 x3 2x 2 mod 17

We want to double the point P 5 1

2P P P 5 1 5 1 x3 y3

x3 s2 x1 x2 5 5 6 mod 17

y3 s x1 x3 y1 6 1 14 3 mod 17

2P 5 1 5 1 6 3

For illustrative purposes we check whether the result 2P 6 3 is actually a point on the curve by inserting the coordinates into the curve equation

y2 x3 2 x 2 mod 2 63 2 6 2 mod 17

Building a Discrete Logarithm Problem with Elliptic Curves

What we have done so far is to establish the group operations point addition and doubling we have provided an identity element and we have shown a way of finding the inverse for any point on the curve Thus we now have all necessary requirements in place to motivate the following theorem

Please note that we have not proved the theorem This theorem is extremely use ful because we have a good understanding of the properties of cyclic groups In particular we know that by definition a primitive element must exist such that its powers generate the entire group Moreover we know quite well how to build cryp tosystems from cyclic groups Here is an example for the cyclic group of an elliptic curve

Example We want to find all points on the curve

E y2 x3 2 x 2 mod 17

It happens that all points on the curve form a cyclic group and that the order is E 19 For this specific curve the group order is a prime and according to Theo rem 4 every element is primitive

As in the previous example we start with the primitive element P 5 1 We compute now all powers of P More precisely since the group operation is addi tion we compute P 2P E P Here is a list of the elements that we obtain

2P 5 1 5 1 6 3 11P 13 10

3P 2P P 10 6 12P 0 11

4P 3 1 13P 16 4

5P 9 16 14P 9 1

6P 16 13 15P 3 16

7P 0 6 16P 10 11

8P 13 7 17P 6 14

9P 7 6 18P 5 16

10P 7 11 19P O

From now on the cyclic structure becomes visible since

20P 19P P O P P

It is also instructive to look at the last computation above which yielded

This means that P 5 1 is the inverse of 18P 5 16 and vice versa This is easy to verify We have to check whether the two x coordinates are identical and that the two y coordinates are each other s additive inverse modulo 17 The first

condition obviously hold and the second one too since

To set up DL cryptosystems it is important to know the order of the group Even though knowing the exact number of points on a curve is an elaborate task we know the approximate number due to Hasse s theorem

Hasse s theorem which is also known as Hasse s bound states that the number of points is roughly in the range of the prime p This has major practical implications For instance if we need an elliptic curve with elements we have to use a prime of length of about bit

Let s now turn our attention to the details of setting up the discrete logarithm problem For this we can strictly proceed as described in Chapter 8

In cryptosystems d is the private key which is an integer while the public key

T is a point on the curve with coordinates T xT yT In contrast in the case of the DL problem in Z p both keys were integers The operation in Eq is called point multiplication since we can formally write T d P This terminology can be

misleading however since we cannot directly multiply the integer d with a curve point P Instead dP is merely a convenient notation for the repeated application of the group operation in Equation 3 Let s now look at an example for an ECDLP

Example We perform a point multiplication on the curve y2 x3 2x 2 mod 17 that was also used in the previous example We want to compute

3 Note that the symbol was chosen arbitrarily to denote the group operation If we had chosen a multiplicative notation instead the ECDLP would have had the form Pd T which would have been more consistent with the conventional DL problem in Z p

13P P P P

where P 5 1 In this case we can simply use the table that was compiled earlier 13P 16 4

Point multiplication is analog to exponentiation in multiplicative groups In or der to do it efficiently we can directly adopt the square and multiply algorithm The only difference is that squaring becomes doubling and multiplication becomes addition of P Here is the algorithm

For a random scalar of length of t 1 bit the algorithm requires on average

t point doubles and additions Verbally expressed the algorithm scans the bit representation of the scalar d from left to right It performs a doubling in every iteration and only if the current bit has the value 1 does it perform an addition of P Let s look at an example

Example We consider the scalar multiplication 26 P which has the following binary representation

26 P 02 P d d d P

The algorithm scans the scalar bits starting on the left with d4 and ending with the rightmost bit d0

Diffie Hellman Key Exchange with Elliptic Curves

0 P 12 P inital setting bit processed d4 1

1a P P 2P P DOUBLE bit processed d3

1b 2P P 3P P 12 P P ADD since d3 1

2a 3P 3P 6P 12 P P DOUBLE bit processed d2

2b no ADD since d2 0

3a 6P 6P 12P 2 P 2 P DOUBLE bit processed d1

3b 12P P 13P 2 P 12 P 2 P ADD since d1 1

4a 13P 13P 26P 2 2 P 02 P DOUBLE bit processed d0

4b no ADD since d0 0

It is instructive to observe how the binary representation of the exponent evolves We see that doubling results in a left shift of the scalar with a 0 put in the rightmost position By performing addition with P a 1 is inserted into the rightmost posi tion of the scalar Compare how the highlighted exponents change from iteration to iteration

If we go back to elliptic curves over the real numbers there is a nice geometric interpretation for the ECDLP given a starting point P we compute 2P 3P

dP T effectively hopping back and forth on the elliptic curve We then publish

the starting point P a public parameter and the final point T the public key In order to break the cryptosystem an attacker has to figure out how often we jumped on the elliptic curve The number of hops is the secret d the private key

Diffie Hellman Key Exchange with Elliptic Curves

In complete analogy to the conventional Diffie Hellman key exchange DHKE in troduced in Sect we can now realize a key exchange using elliptic curves This is referred to as elliptic curve Diffie Hellman key exchange or ECDH First we have to agree on domain parameters that is a suitable elliptic curve over which we can work and a primitive element on this curve

Note that in practice finding a suitable elliptic curve is a relatively difficult task The curves have to show certain properties in order to be secure More about this is said below The actual key exchange is done the same way it was done for the conventional Diffie Hellman protocol

The correctness of the protocol is easy to prove

Proof Alice computes

Since point addition is associative remember that associativity is one of the group properties both parties compute the same result namely the point TAB ab P nu As can be seen in the protocol Alice and Bob choose the private keys a and

b respectively which are two large integers With the private keys both generate their respective public keys A and B which are points on the curve The public keys are computed by point multiplication The two parties exchange these public parameters with each other The joint secret TAB is then computed by both Alice and Bob by performing a second point multiplication involving the public key they received and their own secret parameter The joint secret TAB can be used to derive a session key e g as input for the AES algorithm Note that the two coordinates xAB yAB are not independent of each other Given xAB the other coordinate can be computed by simply inserting the x value in the elliptic curve equation Thus only one of the two coordinates should be used for the derivation of a session key Let s look at an example with small numbers

Example We consider the ECDH with the following domain parameters The elliptic curve is y2 x3 2x 2 mod 17 which forms a cyclic group of order E

19 The base point is P 5 1 The protocol proceeds as follows

choose a kpr A 3 choose b kpr B 10

A kpub A 3 P 10 6 B kpub B 10 P 7 11

TAB aB 3 7 11 13 10 TAB bA 10 10 6 13 10

The two scalar multiplications that each Alice and Bob perform require the Double and Add algorithm

One of the coordinates of the joint secret TAB can now be used as session key In practice often the x coordinate is hashed and then used as a symmetric key Typ ically not all bits are needed For instance in a bit ECC scheme hashing the x coordinate with SHA 1 results in a bit output of which only would be used as an AES key

Please note that elliptic curves are not restricted to the DHKE In fact almost all other discrete logarithm protocols in particular digital signatures and encryption e g variants of Elgamal can also be realized The widely used elliptic curve digital signature algorithms ECDSA will be introduced in Sect 1

The reason we use elliptic curves is that the ECDLP has very good one way char acteristics If an attacker Oscar wants to break the ECDH he has the following information E p P A and B He wants to compute the joint secret between Alice and Bob TAB a b P This is called the elliptic curve Diffie Hellman problem ECDHP There appears to be only one way to compute the ECDHP namely to solve either of the discrete logarithm problems

If the elliptic curve is chosen with care the best known attacks against the

ECDLP are considerably weaker than the best algorithms for solving the DL prob lem modulo p and the best factoring algorithms which are used for RSA attacks In particular the index calculus algorithms which are powerful attacks against the DLP modulo p are not applicable against elliptic curves For carefully selected el liptic curves the only remaining attacks are generic DL algorithms that is Shanks baby step giant step method and Pollard s rho method which were described in Sect 3 Since the number of steps required for such an attack is roughly equal

to the square root of the group cardinality a group order of at least should be used According to Hasse s theorem this requires that the prime p used for the el

liptic curve must be roughly bit long If we attack such a group with generic

algorithms we need around steps A security level of 80 bit provides

medium term security In practice elliptic curve bit lengths up to bit are com monly used which provide security levels of up to bit

It should be stressed that this security is only achieved if cryptographically strong elliptic curves are used There are several families of curves that possess crypto graphic weaknesses e g supersingular curves They are relatively easy to spot however In practice often standardized curves such as ones proposed by the Na tional Institute of Standards and Technology NIST are being used

Implementation in Software and Hardware

Before using ECC a curve with good cryptographic properties needs to be identi fied In practice a core requirement is that the cyclic group or subgroup formed by the curve points has prime order Moreover certain mathematical properties that lead to cryptographic weaknesses must be ruled out Since assuring all these prop erties is a nontrivial and computationally demanding task often standardized curves are used in practice

When implementing elliptic curves it is useful to view an ECC scheme as a struc ture with four layers On the bottom layer modular arithmetic i e arithmetic in the prime field GF p is performed We need all four field operations addition sub traction multiplication and inversion On the next layer the two group operations point doubling and point addition are realized They make use of the arithmetic pro vided in the bottom layer On the third layer scalar multiplication is realized which uses the group operations of the previous layer The top layer implements the actual protocol e g ECDH or ECDSA It is important to note that two entirely different finite algebraic structures are involved in an elliptic curve cryptosystem There is a finite field GF p over which the curve is defined and there is the cyclic group which is formed by the points on the curve

In software a highly optimized bit ECC implementation on a 3 GHz 64 bit CPU can take approximately 2 ms for one point multiplication Slower through puts due to smaller microprocessors or less optimized algorithms are common with performances in the range of 10 ms For high performance applications e g for Internet servers that have to perform a large number of elliptic curve signatures per second hardware implementations are desirable The fastest implementations can compute a point multiplication in the range of 40 s while speeds of several

On the other side of the performance spectrum ECC is the most attractive public key algorithm for lightweight applications such as RFID tags Highly compact ECC engines are possible which need as little as 00 gate equivalences and run at a speed of several tens of milliseconds Even though ECC engines are much larger

Discussion and Further Reading

than implementations of symmetric ciphers such as 3DES they are considerably smaller than RSA implementations

The computational complexity of ECC is cubic in the bit length of the prime used This is due to the fact that modular multiplication which is the main operation on the bottom layer is quadratic in the bit length and scalar multiplication i e with the Double and Add algorithm contributes another linear dimension so that we have in total a cubic complexity This implies that doubling the bit length of an ECC implementation results in performance degradation by a factor of roughly 23 8 RSA and DL systems show the same cubic runtime behavior The advantage of ECC over the other two popular public key families is that the parameters have to be increased much more slowly to enhance the security level For instance doubling the effort of an attacker for a given ECC system requires an increase in the length of the parameter by 2 bits whereas RSA or DL schemes require an increase of 20 30 bits This behavior is due to the fact that only generic attacks cf Sect 3 are known ECC cryptosystems whereas more powerful algorithms are available for attacking RSA and DL schemes

Discussion and Further Reading

History and General Remarks ECC was independently invented in by Neal Koblitz and in by Victor Miller During the s there was much speculation about the security and practicality of ECC especially if compared to RSA After a period of intensive research they appear nowadays very secure just like RSA and DL schemes An important step for building confidence in ECC was the issuing of two ANSI banking standards for elliptic curve digital signature and key establish ment in and respectively 6 7 Interestingly in Suite B a collection of crypto algorithms selected by the NSA for use in US government systems only ECC schemes are allowed as asymmetric algorithms Elliptic curves are also widely used in commercial standards such as IPsec or Transport Layer Security TLS

At the time of writing there still exist far more fielded RSA and DL applications than elliptic curve ones This is mainly due to historical reasons and due to the quite complex patent situation of some ECC variants Nevertheless in many new applica tions with security needs especially in embedded systems such as mobile devices ECC is often the preferred public key scheme For instance ECC is used in the most popular business handheld devices Most likely ECC will become more widespread in the years to come Reference describes the historical development of ECC with respect to scientific and commercial aspects and makes excellent reading

For readers interested in a deeper understanding of ECC the books 25 24 90 44 are recommended The overview article even though a bit dated now provides a good state of the art summary as of the year For more recent de velopments the annual Workshop on Elliptic Curve Cryptography ECC is recom mended as an excellent resource The workshop includes both theoretical and

applied topics related to ECC and related crypto schemes There is also a rich liter ature that deals with the mathematics of elliptic curves regardless of their use in cryptography

Implementation and Variants In the first few years after the invention of ECC these algorithms were believed to be computationally more complex than existing public key schemes especially RSA This assumption is somewhat ironic in hind sight given that ECC tends to be often faster than most other public key schemes During the s fast implementation techniques for ECC was intensively re searched which resulted in considerable performance improvements

In this chapter elliptic curves over prime fields GF p were introduced These

are currently in practice somewhat more widely used than over other finite fields but curves over binary Galois fields GF 2m are also popular For efficient implemen tations improvements are possible at the finite field arithmetic layer at the group operation layer and at the point multiplication layer There is a wealth of techniques and in the following is a summary of the most common acceleration techniques in practice For curves over GF p generalized Mersenne primes are often used at the arithmetic level These are primes such as p 1 Their major advantage is that modulo reduction is extremely simple If general primes are used methods similar to those described in Sect 0 are applicable With respect to ECC over fields GF 2m efficient arithmetic algorithms are described in 90 On the group operation layer several optimizations are possible A popular one is to switch from the affine coordinates that were introduced here to projective coordinates in which each point is represented as a triple x y z Their advantage is that no inversion is required within the group operation The number of multiplications increases however On the next layer fast scalar multiplication techniques are applicable Im proved versions of the Double and Add algorithm which make use of the fact that adding or subtracting a point come at almost identical costs are commonly being applied An excellent compilation of efficient computation techniques for ECC is the book 90

A special type of elliptic curve that allows for particularly fast point multiplica tion is the Koblitz curve These are curves over GF 2m where the coefficients have the values 0 or 1 There have also been numerous other suggestions for ellip tic curves with good implementation properties One such proposal involves elliptic curves over optimum extension fields i e fields of the form GF pm p 2 10

As mentioned in Sect standardized curves are often being used in practice A widely used set of curves is provided in the FIPS Standard Appendix D Alternatives are curves specified by the ECC Brainpool consortium or the Standards for Efficient Cryptography Group SECG 34 9

Elliptic curves also allow for many variants and generalization They are a special case of hyperelliptic curves which can also be used to build discrete logarithm cryp tosystems 44 A summary of implementation techniques for hyperelliptic curves is given in A completely different type of public key scheme which also makes use of elliptic curves is identity based cryptosystems 30 which have drawn much attention over the last few years

Elliptic Curve Cryptography ECC is based on the discrete logarithm problem It requires arithmetic modulo a prime or in a Galois field GF 2m

ECC can be used for key exchange for digital signatures and for encryption

ECC provides the same level of security as RSA or discrete logarithm sys tems over Z p with considerably shorter operands approximately 6 bit vs bit which results in shorter ciphertexts and signatures

In many cases ECC has performance advantages over other public key algo rithms However signature verification with short RSA keys is still considerably faster than ECC

ECC is slowly gaining popularity in applications compared to other public key schemes i e many new applications especially on embedded platforms make use of elliptic curve cryptography

Show that the condition 0 mod p is fulfilled for the curve

y2 x3 2x 2 mod 17

Perform the additions

in the group of the curve y2 x3 2x 2 mod 17 Use only a pocket calculator

In this chapter the elliptic curve y2 x3 2x 2 mod 17 is given with E 19 Verify Hasse s theorem for this curve

Let us again consider the elliptic curve y2 x3 2x 2 mod 17 Why are all

points primitive elements

Note In general it is not true that all elements of an elliptic curve are primitive

Let E be an elliptic curve defined over Z7

E y2 x3 3x 2

Compute all points on E over Z7

What is the order of the group Hint Do not miss the neutral element O

Given the element 0 3 determine the order of Is a primitive element

In practice a and k are both in the range p and computing T a

P and y0 k P is done using the Double and Add algorithm as shown in Sect

Illustrate how the algorithm works for a 19 and for a Do not perform elliptic curve operations but keep P a variable

How many i point additions and ii point doublings are required on average for one multiplication Assume that all integers have n log2 p bit

Assume that all integers have n bit i e p is a bit prime Assume one group operation addition or doubling requires 20 sec What is the time for one double and add operation

Given an elliptic curve E over Z29 and the base point P 8 10

E y2 x3 4x 20 mod 29

Calculate the following point multiplication k P using the Double and Add algo rithm Provide the intermediate results after each step

Given is the same curve as in The order of this curve is known to be E

37 Furthermore an additional point Q 15 P 14 23 on this curve is given Determine the result of the following point multiplications by using as few group operations as possible i e make smart use of the known point Q Specify how you simplified the calculation each time

Hint In addition to using Q use the fact that it is easy to compute P

You should be able to perform the scalar multiplications with considerably fewer steps than a straightforward application of the double and add algorithm would al low

Your task is to compute a session key in a DHKE protocol based on elliptic curves Your private key is a 6 You receive Bob s public key B 5 9 The elliptic curve being used is defined by

y2 x3 x 6 mod 11

An example for an elliptic curve DHKE is given in Sect Verify the two scalar multiplications that Alice performs Show the intermediate results within the group operation

After the DHKE Alice and Bob possess a mutual secret point R x y The modulus of the used elliptic curve is a 64 bit prime Now we want to derive a session key for a bit block cipher The session key is calculated as follows

Describe an efficient brute force attack against the symmetric cipher How many of the key bits are truly random in this case Hint You do not need to describe the mathematical details Provide a list of the necessary steps Assume you have a function that computes square roots modulo p

Derive the formula for addition on elliptic curves That is given the coordi nates for P and Q find the coordinates for R x3 y3

Hint First find the equation of a line through the two points Insert this equation in the elliptic curve equation At some point you have to find the roots of a cubic

polynomial x3 a a1x a0 If the three roots are denoted by x0 x1 x2 you can use the fact that x0 x1 x2 a2

Digital signatures are one of the most important cryptographic tools they and are widely used today Applications for digital signatures range from digital certificates for secure e commerce to legal signing of contracts to secure software updates To gether with key establishment over insecure channels they form the most important instance for public key cryptography

Digital signatures share some functionality with handwritten signatures In par ticular they provide a method to assure that a message is authentic to one user i e it in fact originates from the person who claims to have generated the message How ever they actually provide much more functionality as we ll learn in this chapter

In this chapter you will learn

The principle of digital signatures

Security services that is the specific objectives that can be achieved by a security system

The RSA digital signature scheme

The Elgamal digital signature scheme and two extensions of it the digital signa ture algorithm DSA and the elliptic curve digital signature algorithm ECDSA

C Paar J Pelzl Understanding Cryptography

DOI 0 6 0 Qc Springer Verlag Berlin Heidelberg

In this section we first provide a motivating example why digital signatures are needed and why they must be based on asymmetric cryptography We then develop the principles of digital signatures Actual signature algorithms are introduced in subsequent sections

Odd Colors for Cars or Why Symmetric Cryptography Is Not Sufficient

The crypto schemes that we have encountered so far had two main goals either to encrypt data e g with AES 3DES or RSA encryption or to establish a shared key e g with the Diffie Hellman or elliptic curve key exchange One might be tempted to think that we are now in a position to satisfy any security needs that arise in practice However there are many other security needs besides encryption and key exchange which are in fact termed security services these are discussed in detail in Sect We now discuss a setting in which symmetric cryptography fails to provide a desirable security function

Assume we have two communicating parties Alice and Bob who share a secret key Furthermore the secret key is used for encryption with a block cipher When Alice receives and decrypts a message which makes semantic sense e g the de crypted message is an actual English text she can in many cases conclude that the message was in fact generated by a person with whom he shares the secret key1 If only Alice and Bob know the key they can be reasonably sure that an attacking third party has not changed the message in transit So far we ve always assumed that the bad guy is an external party that we often named Oscar However in practice it is often the case that Alice and Bob do want to communicate securely with each other but at the same time they might be interested in cheating each other It turns out that symmetric key schemes do not protect the two parties against each other Consider the following scenario

Suppose that Alice owns a dealership for new cars where you can select and order cars online We assume that Bob the customer and Alice the dealer have established a shared secret kAB e g by using the Diffie Hellman key exchange Bob now specifies the car that he likes which includes a color choice of pink for the interior and an external color of orange choices most people would not make He sends the order form AES encrypted to Alice She decrypts the order and is happy to have sold another model for 00 Upon delivery of the car three weeks later Bob has second thoughts about his choice in part because his spouse is threatening

1 One has to be a bit careful with such a conclusion though For instance if Alice and Bob use a stream cipher an attacker can flip individual bits of the ciphertext which results in bit flips in the received plaintext Depending on the application the attacker might be able to manipulate the message in a way that is semantically still correct However using block ciphers especially in a chaining mode makes it quite likely that ciphertext manipulations can be detected after decryption

him with divorce after seeing the car Unfortunately for Bob and his family Alice has a no return policy Given that she is an experienced car dealer she knows too well that it will not be easy to sell a pink and orange car and she is thus set on not making any exceptions Since Bob now claims that he never ordered the car she has no other choice but to sue him In front of the judge Alice s lawyer presents Bob s digital car order together with the encrypted version of it Obviously the lawyer argues Bob must have generated the order since he is in possession of kAB with which the ciphertext was generated However if Bob s lawyer is worth his money he will patiently explain to the judge that the car dealer Alice also knows kAB and that Alice has in fact a high incentive to generate faked car orders The judge it turns out has no way of knowing whether the plaintext ciphertext pair was generated by Bob or Alice Given the laws in most countries Bob probably gets away with his dishonesty

This might sound like a rather specific and somewhat artificially constructed sce nario but in fact it is not There are many many situations where it is important to prove to a neutral third party i e a person acting as a judge that one of two or more parties generated a message By proving we mean that the judge can conclude without doubt who has generated the message even if all parties are potentially dis honest Why can t we use some complicated symmetric key scheme to achieve this goal The high level explanation is simple Exactly because we have a sym metric set up Alice and Bob have the same knowledge namely of keys and thus the same capabilities Everything that Alice can do can be done by Bob too Thus a neutral third party cannot distinguish whether a certain cryptographic operation was performed by Alice or by Bob or by both Generally speaking the solution to this problem lies in public key cryptography The asymmetric set up that is inherent in public key algorithms might potentially enable a judge to distinguish between actions that only one person can perform namely the person in possession of the private key and those that can be done by both namely computations involving the public key It turns out that digital signatures are public key algorithms which have the properties that are needed to resolve a situation of cheating participants In the e commerce car scenario above Bob would have been required to digitally sign his order using his private key

Principles of Digital Signatures

The property of proving that a certain person generated a message is obviously also very important outside the digital domain In the real analog world this is achieved by handwritten signatures on paper For instance if we sign a contract or sign a check the receiver can prove to a judge that we actually signed the message Of course one can try to forge signatures but there are legal and social barriers that prevent most people from even attempting to do so As with conventional hand written signatures only the person who creates a digital message must be capable of generating a valid signature In order to achieve this with cryptographic primi

tives we have to apply public key cryptography The basic idea is that the person who signs the message uses a private key and the receiving party uses the matching public key The principle of a digital signature scheme is shown in Fig

Principle of digital signatures which involves signing and verifying a message

The process starts with Bob signing the message x The signature algorithm is a function of Bob s private key kpr Hence assuming he in fact keeps his private key private only Bob can sign a message x on his behalf In order to relate a signature to the message x is also an input to the signature algorithm After signing the message the signature s is appended to the message x and the pair x s is sent to Alice It is important to note that a digital signature by itself is of no use unless it is accom panied by the message A digital signature without the message is the equivalent of a handwritten signature on a strip of paper without the contract or a check that is supposed to be signed

The digital signature itself is merely a large integer value for instance a string of bits The signature is only useful to Alice if she has means to verify whether the signature is valid or not For this a verification function is needed which takes both x and the signature s as inputs In order to link the signature to Bob the function also requires his public key Even though the verification function has long inputs its only output is the binary statement true or false If x was actually signed with the private key that belongs to the public verification key the output is true otherwise it is false

From these general observations we can easily develop a generic digital signature protocol

From this set up the core property of digital signatures follows A signed mes sage can unambiguously be traced back to its originator since a valid signature can only be computed with the unique signer s private key Only the signer has the abil ity to generate a signature on his behalf Hence we can prove that the signing party has actually generated the message Such a proof can even have legal meaning for instance as in the Electronic Signatures in Global and National Commerce Act ES IGN in the USA or in the Signaturgesetz or Signature Law in Germany We note that the basic protocol above does not provide any confidentiality of the message since the message x is being sent in the clear Of course the message can be kept confidential by also encrypting it e g with AES or 3DES

Each of the three popular public key algorithm families namely integer factor ization discrete logarithms and elliptic curves allows us to construct digital signa tures In the remainder of this chapter we learn about most signature schemes that are of practical relevance

It is very instructive to discuss in more detail the security functions we can achieve with digital signatures In fact at this point we will step for a moment away from digital signature and ask ourselves in general What are possible security objectives that a security system might possess More accurately the objectives of a security systems are called security services There exist many security services but the most important ones which are desirable in many applications are as follows

Confidentiality Information is kept secret from all but authorized parties

Integrity Messages have not been modified in transit

Message Authentication The sender of a message is authentic An alternative term is data origin authentication

Nonrepudiation The sender of a message can not deny the creation of the mes sage

Different applications call for different sets of security services For instance for private e mail the first three functions are desirable whereas a corporate e mail sys

tem might also require nonrepudiation As another example if we want to secure software updates for a cell phone the chief objectives might be integrity and mes sage authentication because the manufacturer primarily wants to assure that only original updates are loaded into the handheld device We note that message authen tication always implies data integrity the opposite is not true

The four security services can be achieved in a more or less straightforward man ner with the schemes introduced in this book For confidentiality one uses primarily symmetric ciphers and less frequently asymmetric encryption Integrity and mes sage authentication are provided by digital signatures and message authentication codes which are introduced in 2 Nonrepudiation can be achieved with dig ital signatures as discussed above

In addition to the four core security services there are several other ones

Identification entity authentication Establish and verify the identity of an en tity e g a person a computer or a credit card

Access control Restrict access to the resources to privileged entities

Availability Assures that the electronic system is reliably available

Auditing Provide evidence about security relevant activities e g by keeping logs about certain events

Physical security Provide protection against physical tampering and or re sponses to physical tampering attempts

Anonymity Provide protection against discovery and misuse of identity

Which security services are desired in a given system is heavily application specific For instance anonymity might make no sense for an e mail system since e mails are supposed to have a clearly identifiable sender On the other hand car to car communication systems for collision avoidance one of the many exciting new applications for cryptography we will see in the next ten years or so have a strong need to keep cars and drivers anonymous in order to avoid tracking As a fur ther example in order to secure an operating system access control to certain parts of a computer system is often of paramount importance Most but not all of these advanced services can be achieved with the crypto algorithms from this book How ever in some cases noncryptographic approaches need to be taken For instance availability is often achieved by using redundancy e g running redundant comput ing or storage systems in parallel Such solutions are only indirectly if at all related to cryptography

The RSA Signature Scheme

The RSA signature scheme is based on RSA encryption introduced in Its security relies on the difficulty of factoring a product of two large primes the integer factorization problem Since its first description in in the RSA signature scheme has emerged as the most widely used digital signatures scheme in practice

Schoolbook RSA Digital Signature

Suppose Bob wants to send a signed message x to Alice He generates the same RSA keys that were used for RSA encryption as shown in At the end of the set up he has the following parameters

The actual signature protocol is shown in the following The message x that is being signed is in the range 1 2 n 1

As can be seen from the protocol Bob computes the signature s for a message x by RSA encrypting x with his private key kpr Bob is the only party who can apply kpr and hence the ownership of kpr authenticates him as the author of the signed message Bob appends the signature s to the message x and sends both to

Alice Alice receives the signed message and RSA decrypts s using Bob s public key kpub yielding x If x and xt match Alice knows two important things First the author of the message was in possession of Bob s secret key and if only Bob has

had access to the key it was in fact Bob who signed the message This is called message authentication Second the message has not been changed in transit so that message integrity is given We recall from the previous section that these are two of the fundamental security services which are often needed in practice

Proof We now prove that the scheme is correct i e that the verification process yields a true statement if the message and signature have not been altered during transmission We start from the verification operation se mod n

se xd e xde x mod n

Due to the mathematical relationship between the private and the public key namely that

raising any integer x Zn to the d e th power yields the integer itself again The proof for this was given in Sect nu

The role of the public and the private keys are swapped compared to the RSA encryption scheme Whereas RSA encryption applies the public key to the message x the signature scheme applies the private key kpr On the other side of the commu nication channel RSA encryption requires the use of the private key by the receiver while the digital signature scheme applies the public key for verification

Let s look at an example with small numbers

Example Suppose Bob wants to send a signed message x 4 to Alice The first steps are exactly the same as it is done for an RSA encryption Bob computes his RSA parameters and sends the public key to Alice In contrast to the encryption scheme now the private key is used for signing while the public key is needed to verify the signature

choose p 3 and q 11

xt se 4 mod 33

xt x mod 33 valid signature

compute signature for message

s xd 47 16 mod 33

Alice can conclude from the valid signature that Bob generated the message and that it was not altered in transit i e message authentication and message integrity are given

It should be noted that we introduced a digital signature scheme only In par ticular the message itself is not encrypted and thus there is not confidentiality If this security service is required the message together with the signature should be encrypted e g using a symmetric algorithm like AES

Computational Aspects

First we note that the signature is as long as the modulus n i e roughly log2 n bit As discussed earlier n is typically in the range from to bit Even though such a signature length is not a problem in most Internet applications it can be undesirable in systems that are bandwidth and or energy constrained e g mobile phones

The key generation process is identical to the one we used for RSA encryption which was discussed in detail in To compute and verify the signature the square and multiply algorithm introduced in Sect is used The acceleration techniques for RSA introduced in Sect are also applicable to the digital signa ture scheme Particularly interesting are short public keys e for instance the choice e 1 This makes verification a very fast operation Since in many practical scenarios a message is signed only once but verified many times the fact that ver ification is very fast is helpful This is e g the case in public key infrastructures which use certificates Certificates are signed only once but are verified over and over again every time a user uses his asymmetric keys cf Sect 3

Like in every other asymmetric scheme it must be assured that the public keys are authentic This means that the verifying party in fact has the public key that is associated with the private signature key If an attacker succeeds in providing the verifier with an incorrect public key that supposedly belongs to the signer the attacker can obviously sign messages In order to prevent the attack certificates can be used a topic which is discussed in 3

The first group of attacks attempts to break the underlying RSA scheme by comput ing the private key d The most general of these attacks tries to factor the modulus n into the primes p and q If an attacker succeeds with this she can compute the private key d from e In order to prevent factoring attacks the modulus must be sufficiently large as discussed in Sect In practice bit or more are recommended

Another attack against the schoolbook RSA signature scheme allows an attacker to generate a valid signature for a random message x The attack works as follows

The attacker impersonates Bob i e Oscar claims to Alice that he is in fact Bob Because Alice performs exactly the same computations as Oscar she will verify the signature as correct However by closely looking at Steps 1 and 2 that Oscar performs one sees that the attack is somewhat odd The attacker chooses the signa ture first and then computes the message As a consequence he cannot control the semantics of the message x For instance Oscar cannot generate a message such as Transfer into Oscar s account Nevertheless the fact that an automated verification process does not recognize the forgery is certainly not a desirable feature For this reason schoolbook RSA signature is rarely used in prac tice and padding schemes are applied in order to prevent this and other attacks

RSA Padding The Probabilistic Signature Standard PSS

The attack above can be prevented by allowing only certain message formats Roughly speaking formatting imposes a rule which allows the verifier Alice in our examples to distinguish between valid and invalid messages this is called padding For example a simple formatting rule could specify that all messages x have trailing bits with the value zero or any other specific bit pattern If Oscar chooses signature values s and computes the message x se mod n it is extremely un

likely that x has this specific format If we require a certain value for the trailing bits the chance that x has this format is 00 which is considerably lower than winning any lottery

We now look at a padding scheme which is widely used in practice Note that a padding scheme for RSA encryption was already discussed in Sect The prob abilistic signature scheme RSA PSS is a signature scheme based on the RSA cryptosystem It combines signature and verification with an encoding of the mes sage

Let s have a closer look at RSA PSS Almost always in practice the message it self is not signed directly but rather the hashed version of it Hash functions compute a digital fingerprint of messages The fingerprint has a fixed length say or bit but accepts messages as inputs of arbitrary lengths More about hash functions and the role the play in digital signatures is found in 1

In order to be consistent with the terminology used in standards we denote the message with M rather than with x Figure depicts the encoding procedure which is known as Encoding Method for Signature with Appendix EMSA Proba bilistic Signature Scheme PSS

After the encoding the actual signing operation is applied to the encoded mes sage EM e g

The verification operation then proceeds in a similar way recovery of the salt value and checking whether the EMSA PSS encoding of the message is correct Note that the receiver knows the values of padding1 and padding2 from the standard

The value H in EM is in essence the hashed version of the message By adding a random value salt prior to the second hashing the encoded value becomes proba bilistic As a consequence if we encode and sign the same message twice we obtain different signatures which is a desirable feature

Principle of EMSA PSS encoding

The Elgamal Digital Signature Scheme

The Elgamal signature scheme which was published in is based on the diffi culty of computing discrete logarithms cf Unlike RSA where encryption and digital signature are almost identical operations the Elgamal digital signature is quite different from the encryption scheme with the same name

Schoolbook Elgamal Digital Signature

As with every public key scheme there is a set up phase during which the keys are computed We start by finding a large prime p and constructing a discrete logarithm problem as follows

The public key is now formed by kpub p and the private key by kpr d

Signature and Verification

Using the private key and the parameters of the public key the signature

sigkpr x kE r s

for a message x is computed during the signing process Note that the signature consists of two integers r and s The signing consists of two main steps choosing a random value kE which forms an ephemeral private key and computing the actual signature of x

On the receiving side the signature is verified as verkpub x r s using the public key the signature and the message

In short the verifier accepts a signature r s only if the relation r rs x mod p is satisfied Otherwise the verification fails In order to make sense of the rather arbitrary looking rules for computing the signature parameters r and s as well as the verification it is helpful to study the following proof

Proof We ll prove the correctness of the Elgamal signature scheme More specif ically we show that the verification process yields a true statement if the verifier uses the correct public key and the correct message and if the signature parameters r s were chosen as specified We start with the verification equation

r rs d r kE s mod p

We require that the signature is considered valid if this expression is identical to x

x d r kE s mod p

According to Fermat s Little Theorem the relationship holds if the exponents on both sides of the expression are identical modulo p 1

x dr kE s mod p 1

from which the construction rule of the signature parameters s follows

s x d r kE 1 mod p 1

The condition that gcd kE p 1 1 is required since we have to invert the ephemeral key modulo p 1 when computing s

Let s look at an example with small numbers

Example Again Bob wants to send a message to Alice This time it should be signed with the Elgamal digital signature scheme The signature and verification process is as follows

t r rs 73 22 mod 29

t x mod 29 valid signature

compute signature for message

choose kE 5 note that gcd 5 28 1

s x d r kE 1 10 17

Computational Aspects

The key generation phase is identical to the set up phase of Elgamal encryption which we introduced in Sect 2 Because the security of the signature scheme relies on the discrete logarithm problem p needs to have the properties discussed in Sect 3 In particular it should have a length of at least bits The prime can be generated using the prime finding algorithms introduced in Sect The private key should be generated by a true random number generator The public key requires one exponentiation using the square and multiply algorithm

The signature consists of the pair r s Both have roughly the same bit length as p so that the total length of the package x r s is about three times as long

as only the message x Computing r requires an exponentiation modulo p which can be achieved with the square and multiply algorithm The main operation when computing s is the inversion of kE This can be done using the extended Euclidean algorithm A speed up is possible through precomputing The signer can generate the ephemeral key kE and r in advance and store both values When a message is to be signed they can be retrieved and used to compute s The verifier performs two exponentiations that are again computed with the square and multiply algorithm and one multiplication

First we must make sure that the verifier has the correct public key Otherwise the attack sketched in Sect is applicable Other attacks are described in the following

Computing Discrete Logarithms

The security of the signature scheme relies on the discrete logarithm problem DLP If Oscar is capable of computing discrete logarithms he can compute the private key d from as well as the ephemeral key kE from r With this knowledge he can sign arbitrary messages on behalf of the signer Hence the Elgamal parameters must be chosen such that the DLP is intractable We refer to Sect 3 for a discussion of possible discrete logarithm attacks One of the key requirements is that the prime p should be at least bit long We have also make sure that small subgroup attacks are not possible In order to counter this attack in practice primitive elements are used to generate a subgroup of prime order In such groups all elements are primitive and small subgroups do not exist

Reuse of the Ephemeral Key

If the signer reuses the ephemeral key kE an attacker can easily compute the private key a This constitutes a complete break of the system Here is how the attack works Oscar observes two digital signatures and messages of the form x r s If the two messages x1 and x2 have the same ephemeral key kE Oscar can detect this since the two r values are the same because they were constructed as r1 r2 kE The

two s values are different and Oscar obtains the following two expressions

s1 x1 d r kE 1 mod p 1

s2 x2 d r kE 1 mod p 1

This is an equation system with the two unknowns d which is Bob s private key and the ephemeral key kE By multiplying both equations by kE it becomes a linear system of equations which can be solved easily Oscar simply subtracts the second equation from the first one yielding

s1 s2 x1 x2 kE 1 mod p 1 from which the ephemeral key follows as

If gcd s1 s2 p 1 1 the equation has multiple solutions for kE and Oscar has to verify which is the correct one In any case using kE Oscar can now also compute the private key through either Eq or Eq

d x1 s1kE mod p 1

With the knowledge of the private key d and the public key parameters Oscar can now freely sign any documents on Bob s behalf In order to avoid the attack fresh ephemeral keys stemming from a random number generator should be used for every digital signature

An attack with small numbers is given in the next example

Example Let s assume the situation where Oscar eavesdrops on the following two messages that were previously signed with Bob s private key and that use the same ephemeral key kE

x1 r s1 26 3 26

x2 r s2 13 3 1

Additionally Oscar knows Bob s public key which is given as

p 29 2 7

With this information Oscar is now able to compute the ephemeral key

and finally reveal Bob s private key d

d x1 s1 kE mod p 1

26 26 5 8 19

Existential Forgery Attack

Similar to the case of RSA digital signatures it is also possible that an attacker gen erates a valid signature for a random message x The attacker Oscar impersonates Bob i e Oscar claims to Alice that he is in fact Bob The attack works as follows

The verification yields a true statement because the following holds

dr i d j r j 1 mod p

d r dr ri j 1 mod p

Since the message was constructed as x si mod p 1 the last expression is equal to

which is exactly Alice s condition for accepting the signature as valid

The attacker computes in Step 3 the message x the semantics of which he cannot control Thus Oscar can only compute valid signatures for pseudorandom messages The attack is not possible if the message is hashed which is in practice very often the case Rather than using the message directly for computing the signature one applies a hash function to the message prior to signing i e the signing equation

s h x d r kE 1 mod p 1

The Digital Signature Algorithm DSA

The native Elgamal signature algorithm described in this section is rarely used in practice Instead a much more popular variant is used known as the Digital Signa ture Algorithm DSA It is a federal US government standard for digital signatures DSS and was proposed by the National Institute of Standards and Technology NIST Its main advantages over the Elgamal signature scheme are that the signa ture is only bit long and that some of the attacks that can threaten the Elgamal scheme are not applicable

We introduce here the DSA standard with a bit length of bit Note that longer bit lengths are also possible in the standard

The keys for DSA are computed as follows

The central idea of DSA is that there are two cyclic groups involved One is the large cyclic group Z p the order of which has bit length of bit The second one is in the bit subgroup of Z p This set up yields shorter signatures as we see in the following

In addition to the bit prime p and a bit prime q there are two other bit length combinations possible for the primes p and q According to the latest version of the standard the combinations shown in Table are allowed

If one of the other bit lengths is required only Steps 1 and 2 of the key generation phase have to be adjusted accordingly More about the issue of bit length will be said in Sect below

Bit lengths of important parameters of DSA

Signature and Verification

As in the Elgamal signature scheme the DSA signature consists of a pair of integers r s Since each of the two parameters is only bit long the total signature length is bit Using the public and private key the signature for a message x is computed as follows

According to the standard the message x has to be hashed using the hash function SHA 1 in order to compute s Hash functions including SHA 1 are described in 1 For now it is sufficient to know that SHA 1 compresses x and computes a bit fingerprint This fingerprint can be thought of as a representative of x

The signature verification process is as follows

The verifier accepts a signature r s only if v r mod q is satisfied Otherwise the verification fails In this case the message or the signature may have been mod ified or the verifier is not in possession of the correct public key In any case the signature should be considered invalid

Proof We show that a signature r s satisfies the verification condition v r mod

q We ll start with the signature parameter s

which is equivalent to

s SHA x d r kE 1 mod q

kE s 1 SHA x d s 1 r mod q

The right hand side can be expressed in terms of the auxiliary values u1 and u2

kE u1 d u2 mod q

We can raise to either side of the equation if we reduce modulo p

kE mod p u1 d u2 mod p

Since the public key value was computed as d mod p we can write

kE mod p u1 u2 mod p

We now reduce both sides of the equation modulo q

kE mod p mod q u1 u2 mod p mod q

Since r was constructed as r kE mod p mod q and v u1 u2 mod p mod q this expression is identical to the condition for verifying a signature as valid

Let s look at an example with small numbers

Example Bob wants to send a message x to Alice which is to be signed with the DSA algorithm Suppose the hash value of x is h x 26 Then the signature and verification process is as follows

choose private key d 7

u1 6 26 11 mod 29

u2 6 20 4 mod 29

v 44 mod 59 mod 29 20

v r mod 29 valid signature

compute hash of message h x 26

choose ephemeral key kE 10

r mod 59 20 mod 29

s 26 7 20 3 5 mod 29

In this example the subgroup has a prime order of q 29 whereas the large cyclic group modulo p has 58 elements We note that 58 9 We replaced the function SHA x by h x since the SHA hash function has an output of length bit

Computational Aspects

We discuss now the computations involved in the DSA scheme The most demand ing part is the key generation phase However this phase only has to be executed once at set up time

The challenge in the key generation phase is to find a cyclic group Z p with a bit length of and which has a prime subgroup in the range of This condi tion is fulfilled if p 1 has a prime factor q of bit The general approach to generating such parameters is to first find the bit prime q and then to construct the larger prime p from it Below is the prime generating algorithm Note that the NIST specified scheme is slightly different

The choice of 2q as modulus in step assures that the prime candidates gener ated in step are odd numbers Since p 1 is divisible by 2q it is also divisible by q If p is a prime Z p thus has a subgroup of order q

During signing we compute the parameters r and s Computing r involves first eval uation gkE mod p using the square and multiply algorithm Since kE has only bit about squarings and multiplications are required on average even though the arithmetic is done with bit numbers The result which has also a length of bit is then reduced to bit by the operation mod q Com puting s involves only bit numbers The most costly step is the inversion of kE

Of these operations the exponentiation is the most costly one in terms of com putational complexity Since the parameter r does not depend on the message it can be precomputed so that the actual signing can be a relatively fast operation

Computing the auxiliary parameters w u1 and u2 only involves bit operands which makes them relatively fast

An interesting aspect of DSA is that we have to protect against two different discrete logarithm attacks If an attacker wants to break DSA he could attempt to compute the private key d by solving the discrete logarithm in the large cyclic group modulo p

The most powerful method for this is the index calculus attack which was sketched in Sect 3 In order to thwart this attack p must be at least bit long It is estimated that this provides a security level of 80 bit i e an attack would need on the order of operations cf in For higher security levels NIST allows primes with lengths of and bit

The second discrete logarithm attack on DSA is to exploit the fact that gen

erates only a small subgroup of order q Hence it seems promising to attack only the subgroup which has a size of about rather than the large cyclic group with about 4 elements formed by p However it turns out that the powerful index calculus attack is not applicable if Oscar wants to exploit the subgroup property The best he can do is to perform one of the generic DLP attacks i e either the baby step giant step method or Pollard s rho method cf Sect 3 These are so called

square root attacks and given that the su bgroup has an order of approximately

these attacks provide a security level of It is not a coincidence that the

index calculus attack and the square root attack have a comparable complexity in fact the parameter sizes were deliberately chosen that way One has to be careful

though if the size of p is increased to or bit This only increases the dif ficulty of the index calculus attack but the small subgroup attack would still have a complexity of if the subgroup stays the same size For this reason q also must be increased if larger p values are chosen Table shows the NIST specified lengths of the primes p and q together with the resulting security levels The security level of the hash function must also match the one of the discrete logarithm problem Since the cryptographic strength of a hash function is mainly determined by the bit length of the hash output the minimum hash output is also given in the table More about security of hash functions will be said in 1

Standardized parameter bit lengths and security levels for DSA

It should be stressed that the record for discrete logarithm calculations is bit so that the bit DSA variant is currently secure and the bit and bit variants seem to provide good long term security

In addition to discrete logarithm attacks DSA becomes vulnerable if the ephe meral key is reused This attack is completely analogues to the case of Elgamal digital signature Hence it must be assured that a fresh randomly genererated key kE is used in every signing operation

The Elliptic Curve Digital Signature Algorithm ECDSA

As discussed in elliptic curves have several advantages over RSA and over DL schemes like Elgamal or DSA In particular in absence of strong attacks against elliptic curve cryptosystems ECC bit lengths in the range of 6 bit can be chosen which provide security equivalent to bit RSA and DL schemes The shorter bit length of ECC often results in shorter processing time and in shorter signatures For these reasons the Elliptic Curve Digital Signature Algo rithm ECDSA was standardized in the US by the American National Standards Institute ANSI in

The steps in the ECDSA standard are conceptionally closely related to the DSA scheme However its discrete logarithm problem is constructed in the group of

an elliptic curve Thus the arithmetic to be performed for actually computing an ECDSA signature is entirely different from that used for DSA

The ECDSA standard is defined for elliptic curves over prime fields Zp and Ga

lois fields GF 2m The former is often preferred in practice and we will only in troduce this one in what follows

The keys for the ECDSA are computed as follows

Note that we have set up a discrete logarithm problem where the integer d is the private key and the result of the scalar multiplication point B is the public key Similar to DSA the cyclic group has an order q which should have a size of at least bit or more for higher security levels

Signature and Verification

Like DSA an ECDSA signature consists of a pair of integers r s Each value has the same bit length as q which makes for fairly compact signatures Using the public and private key the signature for a message x is computed as follows

In step 3 the x coordinate of the point R is assigned to the variable r The mes sage x has to be hashed using the function h in order to compute s The hash function output length must be at least as long as q More about the choice of the hash func tion will be said in 1 However for now it is sufficient to know that the hash function compresses x and computes a fingerprint which can be viewed as a representative of x

The signature verification process is as follows

In the last step the notation xP indicates the x coordinate of the point P The verifier accepts a signature r s only if the xP has the same value as the signature parameter r modulo q Otherwise the signature should be considered invalid

Proof We show that a signature r s satisfies the verification condition r xP mod

q We ll start with the signature parameter s

s h x d r kE 1 mod q

which is equivalent to

kE s 1 h x d s 1 r mod q

The right hand side can be expressed in terms of the auxiliary values u1 and u2

kE u1 d u2 mod q

Since the point A generates a cyclic group of order q we can multiply both sides of the equation with A

kE A u1 d u2 A

Since the group operation is associative we can write

What we showed so far is that the expression u1 A u2 B is equal to kE A if the correct signature and key and message have been used But this is exactly the condition that we check in the verification process by comparing the x coordinates of P u1 A u2 B and R kE A

Using the small elliptic curve from we look at a simple ECDSA exam ple

Example Bob wants to send a message to Alice that is to be signed with the ECDSA algorithm The signature and verification process is as follows

choose E with p 17 a 2 b

A 5 1 with q 19 choose d 7

compute B dA 7 5 1

u1 9 26 6 mod 19

u2 9 7 6 mod 19

P 6 5 1 6 0 6 7 11

xP r mod 19 valid signature

2 2 5 1 0 6

compute hash of message h x

choose ephemeral key kE 10

R 10 5 1 7 11

s 26 7 7 2 17 mod 19

Note that we chose the elliptic curve

E y2 x3 2x 2 mod 17

which is discussed in Sect Because all points of the curve form a cyclic group of order 19 i e a prime there are no subgroups and hence in this case q E 19

Computational Aspects

We discuss now the computations involved in the three stages of the ECDSA scheme

Key Generation As discussed earlier finding an elliptic curve with good crypto graphic properties is a nontrivial task In practice standardized curves such as the ones proposed by NIST or the Brainpool consortium are often used The remaining computation in the key generation phase is one point multiplication which can be done using the double and add algorithm

Signing During signing we first compute the point R which requires one point multiplication and from which r immediately follows For the parameter s we have to invert the ephemeral key which is done with the extended Euclidean algorithm The other main operations are hashing of the message and one reduction modulo q The point multiplication which is in most cases by the far the most arithmetic intensive operation can be precomputed by choosing the ephemeral key ahead of time e g during the idle time of a CPU Thus in situations where precomputation

is an option signing becomes a very fast operation

Verification Computing the auxiliary parameters w u1 and u2 involves straightfor ward modular arithmetic The main computational load occurs during the evaluation of Pu1 A u2 B This can be accomplished by two separate point multiplications However there are specialized methods for simultaneous exponentiations remem ber from that point multiplication is closely related to exponentiation which are faster than two individual point multiplications

Given that the elliptic curve parameters are chosen correctly the main analytical at tack against ECDSA attempts to solve the elliptic curve discrete logarithm problem If an attacker were capable of doing this he could compute the private key d and or the ephemeral key However the best known ECC attacks have a complexity propor

tional to the square root of the size of the group in which the DL problem is defined i e proportional to q The parameter length of ECDSA and the corresponding

security levels are given in Table We recall that the prime p is typically only slightly larger than q so that all arithmetic for ECDSA is done with operands which have roughly the bit length of q

The security level of the hash function must also match that of the discrete loga rithm problem The cryptographic strength of a hash function is mainly determined by the length of its output More about security of hash functions will be said in 1

The security levels of and were chosen so that they match the security offered by AES with its three respective key sizes

More subtle attacks against ECDSA are also possible For instance at the begin ning of verification it must be checked whether r s 1 2 q in order to prevent a certain attack Also protocol based weaknesses e g reusing the ephemeral key must be prevented

Discussion and Further Reading

Bit lengths and security levels of ECDSA

Discussion and Further Reading

Digital Signature Algorithms The first practical realization of digital signatures was introduced in the original paper by Rivest Shamir and Adleman RSA digital signatures have been standardized by several bodies for a long time see e g 95 RSA signatures were and in many cases still are the de facto standard for many applications especially for certificates on the Internet

The Elgamal digital signature was published in in 73 Many variants of this scheme are possible and have been proposed over the years For a compact summary see Note 0

The DSA algorithm was proposed in and became a US standard in There were two possible motivations for the government to create this standard as an alternative to RSA First RSA was patented at that time and having a free alternative was attractive for US industry Second an RSA digital signature implementation can also be used for encryption This was not desirable from the US government viewpoint since there were still rather strict export restrictions for cryptography in the US at that time In contrast a DSA implementation can only be used for signing and not for encryption and it was easier to export systems that only included signature functionality Note that DSA refers to the digital signature algorithm and the corresponding standard is referred to as DSS the digital signature standard Today DSS includes not only the DSA algorithm but also ECDSA and RSA digital signatures

In addition to the algorithms discussed in this chapter there exist several other schemes for digital signatures These include e g the Rabin signature the Fiat Shamir signature 76 the Pointcheval Stern signature and the Schnorr signature

Using Digital Signatures With digital signatures the problem of authentic public keys is acute How can Alice or Bob assure that they possess the correct public keys for each other Or phrased differently how can Oscar be prevented from in jecting faked public keys in order to perform an attack We discuss this question in detail in 3 where certificates are introduced Certificates are based on digital signatures and are one of the main applications of digital signatures They bind an identity e g Alice s e mail address to a public key

One of the more interesting interactions between society and cryptography is digital signature laws They basically assure that a cryptographic digital signature has a legally binding meaning For instance an electronic contract that was digitally

signed can be enforced in the same way as a conventionally signed contract Around the turn of the millennium many nations introduced corresponding laws This was at a time that the brave new world of the Internet had opened up seemingly endless opportunities for doing business online and digital signature laws seemed to be crucial to allow trusted business transactions via the Internet Examples of digital signature laws are the Electronic Signatures in Global and National Commerce Act ESIGN in the US or the corresponding directive of the European Union A good online source for more information is the Digital Law Survey Even though much electronic commerce is today conducted without making use of signature laws there will be without doubt more and more situations where those laws are actually needed

One crucial issue when using digital signatures in the real world is that the private keys especially if used in a setting with legal significance have to be kept strictly confidential This requires a secure way to store this delicate key material One way to satisfy this requirement is to employ smart cards that can be used as secure containers for secret keys A secret key never leaves the smart card and signatures are performed within the CPU inside the smart card For applications with high security requirements so called tamper resistant smart cards are protected against several types of hardware attacks Reference provides excellent insight into the various facets of the highly sophisticated smart card technology

Digital signatures provide message integrity message authentication and nonre pudiation

One of the main application areas of digital signatures is certificates

RSA is currently the most widely used digital signature algorithm Competitors are the Digital Signature Standard DSA and the Elliptic Curve Digital Signature Standard ECDSA

The Elgamal signature scheme is the basis for DSA In turn ECDSA is a gener alization of DSA to elliptic curves

RSA verification can be done with short public keys e Hence in practice RSA verification is usually faster than signing

DSA and ECDSA have the advantage over RSA in that the signatures are much shorter

In order to prevent certain attacks RSA should be used with padding

The modulus of DSA and the RSA signature schemes should be at least bits long For true long term security a modulus of length bits should be chosen In contrast ECDSA achieves the same security levels with bit lengths in the range 6 bits

In Sect we state that sender or message authentication always implies data integrity Why Is the opposite true too i e does data integrity imply sender authentication Justify both answers

In this exercise we want to consider some basic aspects of security services

Does privacy always guarantee integrity Justify your answer

In which order should confidentiality and integrity be assured should the entire message be encrypted first or last Give the rationale for your answer

Design a security service that provides data integrity data confidentiality and nonrepudiation using public key cryptography in a two party communication sys tem over an insecure channel Give a rationale that data integrity confidentiality and nonrepudiation are achieved by your solution Recommendation Consider the corresponding threats in your argumentation

A painter comes up with a new business idea He wants to offer custom paint ings from photos Both the photos and paintings will be transmitted in digital form via the Internet One concern that he has is discretion towards his customers since potentially embarrassing photos e g nude photos might be sent to him Hence the photo data should not be accessible for third parties during transmission The painter needs multiple weeks for the creation of a painting and hence he wants to assure that he cannot be fooled by someone who sends in a photo assuming a false name He also wants to be assured that the painting will definitely be accepted by the customer and that she cannot deny the order

Choose the necessary security services for the transmission of the digitalized photos from the customers to the painter

Which cryptographic elements e g symmetric encryption can be utilized to achieve the security services Assume that several megabytes of data have to be transmitted for every photo

Given an RSA signature scheme with the public key n e which of the following signatures are valid

Given an RSA signature scheme with the public key n e show how Oscar can perform an existential forgery attack by providing an example of such for the parameters of the RSA digital signature scheme

In an RSA digital signature scheme Bob signs messages xi and sends them together with the signatures si and her public key to Alice Bob s public key is the pair n e her private key is d

Oscar can perform man in the middle attacks i e he can replace Bob s public key by his own on the channel His goal is to alter messages and provide these with a digital signature which will check out correctly on Alice s side Show everything that Oscar must do for a successful attack

Given is an RSA signature scheme with EMSA PSS padding as shown in Sect Describe the verification process step by step that has to be performed by the receiver of a signature that was EMSA PSS encoded

One important aspect of digital signatures is the computational effort required to i sign a message and ii to verify a signature We study the computational complexity of the RSA algorithm used as a digital signature in this problem

How many multiplications do we need on average to perform i signing of a message with a general exponent and ii verification of a signature with the short exponent e 1 Assume that n has l log2 n bits Assume the square and multiply algorithm is used for both signing and verification Derive general expressions with l as a variable

Which takes longer signing or verification

We now derive estimates for the speed of actual software implementation Use the following timing model for multiplication The computer operates with 32 bit data structures Hence each full length variable in particular n and x is repre sented by an array with m l 32 elements with x being the basis of the ex ponentiation operation We assume that one multiplication or squaring of two of these variables modulo n takes m2 time units a time unit is the clock period times some constant larger than one which depends on the implementation Note that you never multiply with the exponents d and e That means the bit length of the exponent does not influence the time it takes to perform an individual modular squaring or multiplication

How long does it take to compute a signature verify a signature if the time unit on a certain computer is nsec and n has bits How long does it take if n has bit

Smart cards are one very important platform for the use of digital signatures Smart cards with an microprocessor kernel are popular in practice The is an 8 bit processor What time unit is required in order to perform one signature generation in sec if n has i bits and ii bits Since these processors cannot be clocked at more than say 10 MHz is the required time unit realistic

We now consider the Elgamal signature scheme You are given Bob s pri vate key Kpr d 67 and the corresponding public key Kpub p 97 23 15

Calculate the Elgamal signature r s and the corresponding verification for a message from Bob to Alice with the following messages x and ephemeral keys kE

You receive two alleged messages x1 x2 with their corresponding signatures ri si from Bob Verify whether the messages x1 r1 s1 22 37 33 and x2 r2 s2 82 13 65 both originate from Bob

Compare the RSA signature scheme with the Elgamal signature scheme Where are their relative advantages and drawbacks

Given is an Elgamal signature scheme with p 31 3 and 6 You receive the message x 10 twice with the signatures r s

Are both signatures valid

How many valid signatures are there for each message x and the specific param eters chosen above

Given is an Elgamal signature scheme with the public parameters p 97 23 15 Show how Oscar can perform an existential forgery attack by providing an example for a valid signature

Given is an Elgamal signature scheme with the public parameters p

Z p and an unknown private key d Due to faulty implementation the following dependency between two consecutive ephemeral keys is fulfilled

Furthermore two consecutive signatures to the plaintexts x1 and x2

are given Explain how an attacker is able to calculate the private key with the given values

The parameters of DSA are given by p 59 q 29 3 and Bob s pri vate key is d 23 Show the process of signing Bob and verification Alice for following hash values h x and ephemeral keys kE

Show how DSA can be attacked if the same ephemeral key is used to sign two different messages

The parameters of ECDSA are given by the curve E y2 x3 2x 2 mod 17 the point A 5 1 of order q 19 and Bob s private d 10 Show the process of signing Bob and verification Alice for following hash values h x and ephemeral keys kE

Hash functions are an important cryptographic primitive and are widely used in protocols They compute a digest of a message which is a short fixed length bit string For a particular message the message digest or hash value can be seen as the fingerprint of a message i e a unique representation of a message Unlike all other crypto algorithms introduced so far in this book hash functions do not have a key The use of hash functions in cryptography is manifold Hash functions are an essential part of digital signature schemes and message authentication codes as discussed in Chapter 12 Hash functions are also widely used for other cryptographic applications e g for storing of password hashes or key derivation

In this chapter you will learn

Why hash functions are required in digital signature schemes

Important properties of hash functions

A security analysis of hash functions including an introduction to the birthday paradox

An overview of different families of hash functions

How the popular hash function SHA 1 works

C Paar J Pelzl Understanding Cryptography

DOI 0 6 1 Qc Springer Verlag Berlin Heidelberg

Motivation Signing Long Messages

Even though hash functions have many applications in modern cryptography they are perhaps best known for the important role they play in the practical use of digital signatures In the previous chapter we have introduced signature schemes based on the asymmetric algorithms RSA and the discrete logarithm problem For all schemes the length of the plaintext is limited For instance in the case of RSA the message cannot be larger than the modulus which is in practice often between and bits long Remember this translates into only 4 bytes most emails are longer than that Thus far we have ignored the fact that in practice the plaintext x will often be much larger than those sizes The question that arises at this point is simple How are we going to efficiently compute signatures of large messages An intuitive approach would be similar to the ECB mode for block ci phers Divide the message x into blocks xi of size less than the allowed input size of the signature algorithm and sign each block separately as depicted in Figure

Insecure approach to signing of long messages

However this approach yields three serious problems

Problem 1 High Computational Load Digital signatures are based on computa tionally intensive asymmetric operations such as modular exponentiations of large integers Even if a single operation consumes a small amount of time and energy which is relevant in mobile applications the signatures of large messages e g email attachments or multimedia files would take too long on current computers Furthermore not only does the signer have to compute the signature but the verifier also has to spend a similar amount of time and energy to verify the signature

Problem 2 Message Overhead Obviously this na ve approach doubles the mes sage overhead because not only must the message be sent but also the signature which is of the same length in this case For instance a 1 MB file must yield an RSA signature of length 1 MB so that a total of 2 MB must be transmitted

Problem 3 Security Limitations This is the most serious problem if we attempt to sign a long message by signing a sequence of message blocks individually The approach shown in Fig leads immediately to new attacks For instance Oscar could remove individual messages and the corresponding signatures or he could re order messages and signatures or he could reassemble new messages and signatures out of fragments of previous messages and signatures etc Even though an attacker

Motivation Signing Long Messages

cannot perform manipulations within an individual block we do not have protection for the whole message

Hence for performance as well as for security reasons we would like to have one short signature for a message of arbitrary length The solution to this problem is hash functions If we had a hash function that somehow computes a fingerprint of the message x we could perform the signature operation as shown in Figure

Signing of long messages with a hash function

Assuming we possess such a hash function we now describe a basic protocol for a digital signature scheme with a hash function Bob wants to send a digitally signed message to Alice

Bob computes the hash of the message x and signs the hash value z with his private key kpr B On the receiving side Alice computes the hash value zt of the received message x She verifies the signature s with Bob s public key kpub B We

note that both the signature generation and the verification operate on the hash value z rather than on the message itself Hence the hash value represents the message The hash is sometimes referred to as the message digest or the fingerprint of the message

Before we discuss the security properties of hash functions in the next section we can now get a rough feeling for a desirable input output behavior of hash func tions We want to be able to apply a hash function to messages x of any size and

it is thus desirable that the function h is computationally efficient Even if we hash large messages in the range of say hundreds of megabytes it should be relatively fast to compute Another desirable property is that the output of a hash function is of fixed length and independent of the input length Practical hash functions have output lengths between 2 bits Finally the computed fingerprint should be highly sensitive to all input bits That means even if we make minor modifications to the input x the fingerprint should look very different This behavior is similar to that of block ciphers The properties which we just described are symbolized in Figure

message message digest

Principal input output behavior of hash functions

Security Requirements of Hash Functions

As mentioned in the introduction unlike all other crypto algorithms we have dealt with so far hash functions do not have keys The question is now whether there are any special properties needed for a hash function to be secure In fact we have to ask ourselves whether hash functions have any impact on the security of an ap plication at all since they do not encrypt and they don t have keys As is often the case in cryptography things can be tricky and there are attacks which use weak nesses of hash functions It turns out that there are three central properties which hash functions need to possess in order to be secure

preimage resistance or one wayness

second preimage resistance or weak collision resistance

collision resistance or strong collision resistance

These three properties are visualized in Figure They are derived in the fol lowing

h x preimage resistance

second preimage resistance

h x1 h x2 collision resistance

The three security properties of hash functions

Preimage Resistance or One Wayness

Hash functions need to be one way Given a hash output z it must be computation ally infeasible to find an input message x such that z h x In other words given a fingerprint we cannot derive a matching message We demonstrate now why preim age resistance is important by means of a fictive protocol in which Bob is encrypting the message but not the signature i e he transmits the pair

ek x sigkpr B z

Here ek is a symmetric cipher e g AES with some symmetric key shared by Alice and Bob Let s assume Bob uses an RSA digital signature where the signature is computed as

The attacker Oscar can use Bob s public key to compute

If the hash function is not one way Oscar can now compute the message x from h 1 z x Thus the symmetric encryption of x is circumvented by the signature which leaks the plaintext For this reason h x should be a one way function

In many other applications which make use of hash functions for instance in key derivation it is even more crucial that they are preimage resistant

Second Preimage Resistance or Weak Collision Resistance

For digital signatures with hash it is essential that two different messages do not hash to the same value This means it should be computationally infeasible to create two different messages x1 x2 with equal hash values z1 h x1 h x2 z2 We differentiate between two different types of such collisions In the first case x1

is given and we try to find x2 This is called second preimage resistance or weak collision resistance The second case is given if an attacker is free to choose both x1 and x2 This is referred to as strong collision resistance and is dealt with in the subsequent section

It is easy to see why second preimage resistance is important for the basic signature with hash scheme that we introduced above Assume Bob hashes and signs a message x1 If Oscar is capable of finding a second message x2 such that h x1 h x2 he can run the following substitution attack

verkpub B s z true

As we can see Alice would accept x2 as a correct message since the verification gives her the statement true How can this happen From a more abstract view point this attack is possible because both signing by Bob and verifying by Alice do not happen with the actual message itself but rather with the hashed version of it Hence if an attacker manages to find a second message with the same fingerprint i e hash output signing and verifying are the same for this second message

The question now is how we can prevent Oscar from finding x2 Ideally we would like to have a hash function for which weak collisions do not exist This is unfor tunately impossible due to the pigeonhole principle a more impressive term for which is Dirichlet s drawer principle The pigeonhole principle uses a counting ar gument in situations like the following If you are the owner of pigeons but in your pigeon loop are only 99 holes at least one pigeonhole will be occupied by 2 birds Since the output of every hash function has a fixed bit length say n bit there are only 2n possible output values At the same time the number of inputs to the hash functions is infinite so that multiple inputs must hash to the same output value In practice each output value is equally likely for a random input so that weak collisions exist for all output values

Since weak collisions exist in theory the next best thing we can do is to assure that they cannot be found in practice A strong hash function should be designed such that given x1 and h x1 it is impossible to construct x2 such that h x1 h x2 This means there is no analytical attack However Oscar can always randomly pick x2 values compute their hash values and check whether they are equal to h x1 This is similar to an exhaustive key search for a symmetric cipher In order to prevent this attack given today s computers an output length of n 80 bit is sufficient However we see in the next section that more powerful attacks exist which force us to use even longer output bit lengths

Collision Resistance and the Birthday Attack

We call a hash function collision resistant or strong collision resistant if it is com putationally infeasible to find two different inputs x1 x2 with h x1 h x2 This property is harder to achieve than weak collision resistance since an attacker has two degrees of freedom Both messages can be altered to achieve similar hash values We show now how Oscar could turn his ability to find collisions into an attack He starts with two messages for instance

x1 Transfer 10 into Oscar s account

x2 Transfer 00 into Oscar s account

He now alters x1 and x2 at nonvisible locations e g he replaces spaces by tabs adds spaces or return signs at the end of the message etc This way the semantics of the message is unchanged e g for a bank but the hash value changes for every version of the message Oscar continues until the condition h x1 h x2 is fulfilled Note that if an attacker has e g 64 locations that he can alter or not this yields versions of the same message with different hash values With the two messages he can launch the following attack

verkpub B s z true

This attack assumes that Oscar can trick Bob into signing the message x1 This is of course not possible in every situation but one can imagine scenarios where Oscar can pose as an innocent party e g an e commerce vendor on the Internet and x1 is the purchase order that is generated by Oscar

As we saw earlier due to the pigeonhole principle collisions always exist The question is how difficult it is to find them Our first guess is probably that this is as difficult as finding second preimages i e if the hash function has an output length of 80 bits we have to check about messages However it turns out that an attacker needs only about messages This is a quite surprising result which is due to the birthday attack This attack is based on the birthday paradox which is a powerful tool that is often used in cryptanalysis

It turns out that the following real world question is closely related to finding collisions for hash functions How many people are needed at a party such that there is a reasonable chance that at least two people have the same birthday By

birthday we mean any of the days of the year Our intuition might lead us to assume that we need around people i e about half the number of days in a year for a collision to occur However it turns out that we need far fewer people The piecewise approach to solve this problem is to first compute the probability of two people not having the same birthday i e having no collision of their birthdays For one person the probability of no collision is 1 which is trivial since a single birthday cannot collide with anyone else s For the second person the probability of no collision is over since there is only one day the birthday of the first person to collide with

P no collision among 2 people 1 1

If a third person joins the party he or she can collide with both of the people already there hence

P no collision among 3 people r1 1 r1 2

Consequently the probability for t people having no birthday collision is given by

P no collision among t people r1 1 r1 2 r1 t 1

For t people we will have a collision with probability 1 since a year has only days We return now to our initial question how many people are needed to have a 50 chance of two colliding birthdays Surprisingly following from the equations above it only requires 23 people to obtain a probability of about for a birthday collision since

P at least one collision 1 P no collision

1 r1 1 r1 23 1

Note that for 40 people the probability is about 90 Due to the surprising outcome of this gedankenexperiment it is often referred to as the birthday paradox

Collision search for a hash function h is exactly the same problem as finding

birthday collisions among party attendees For a hash function there are not values each element can take but 2n where n is the output width of h In fact it turns out that n is the crucial security parameter for hash functions The question is how many messages x1 x2 xt does Oscar need to hash until he has a reasonable chance that h xi h xj for some xi and x j that he picked The probability for no collisions among t hash values is

P no collision r1 1 r1 2 r1 t 1

We recall from our calculus courses that the approximation

holds1 since i 2n 1 We can approximate the probability as

The arithmetic series

1 2 t 1 t t 1 2

is in the exponent which allows us to write the probability approximation as

P no collision e t t 1

Recall that our goal is to find out how many messages x1 x2 xt are needed to find a collision Hence we solve the equation now for t If we denote the probability of at least one collision by 1 P no collision then

t t 1 2n 1 lnr 1

Since in practice t 1 it holds that t2 t t 1 and thus

1 This follows from the Taylor series representation of the exponential function e x 1 x

x x for x 1

Equation is extremely important it describes the relationship between the number of hashed messages t needed for a collision as a function of the hash output length n and the collision probability The most important consequence of the birthday attack is that the number of messages we need to hash to find a colli

sion is roug hly equal to the square root of the number of possible output values

i e about 2n 2n 2 Hence for a security level cf Section 4 of x bit the

hash function needs to have an output length of 2x bit As an example assume we want to find a collision for a hypothetical hash function with 80 bit output For a success probability of 50 we expect to hash about

t 2 ln 1 1 2

input values Computing around hashes and checking for collisions can be done with current laptops In order to thwart collision attacks based on the birthday para dox the output length of a hash function must be about twice as long as an output length which protects merely against a second preimage attack For this reason all hash functions have an output length of at least bit where most modern ones are much longer Table shows the number of hash computations needed for a birthday paradox collision for output lengths found in current hash functions Interestingly the desired likelihood of a collision does not influence the attack com plexity very much as is evidenced by the small difference between the success probabilities and It should be stressed that the birthday attack is a

Number of hash values needed for a collision for different hash function output lengths and for two different collision likelihoods

generic attack This means it is applicable against any hash function On the other hand it is not guaranteed that it is the most powerful attack available for a given hash function As we will see in the next section for some of the most popular hash functions in particular MD5 and SHA 1 mathematical collision attacks exist which are faster than the birthday attack

It should be stressed that there are many applications for hash functions e g storage of passwords which only require preimage resistance Thus a hash function with a relatively short output say 80 bit might be sufficient since collision attacks do not pose a threat

At the end of this section we summarize all important properties of hash functions

h x Note that the first three are practical requirements whereas the last three relate to the security of hash functions

Overview of Hash Algorithms

So far we only discussed the requirements for hash functions We now introduce how to actually built them There are two general types of hash functions

Dedicated hash functions These are algorithms that are specifically designed to serve as hash functions

Block cipher based hash functions It is also possible to use block ciphers such as AES to construct hash functions

As we saw in the previous section hash functions can process an arbitrary length message and produce a fixed length output In practice this is achieved by segment ing the input into a series of blocks of equal size These blocks are processed se quentially by the hash function which has a compression function at its heart This iterated design is known as Merkle Damga rd construction The hash value of the input message is then defined as the output of the last iteration of the compression function Fig

Merkle Damga rd hash function construction

Dedicated Hash Functions The MD4 Family

Dedicated hash functions are algorithms that have been custom designed A large number of such constructions have been proposed over the last two decades In prac tice by far the most popular ones have been the hash functions of what is called the MD4 family MD5 the SHA family and RIPEMD are all based on the principles of MD4 MD4 is a message digest algorithm developed by Ronald Rivest MD4 was an innovative idea because it was especially designed to allow very efficient soft ware implementation It uses 32 bit variables and all operations are bitwise Boolean functions such as logical AND OR XOR and negation All subsequent hash func tions in the MD4 family are based on the same software friendly principles

A strengthened version of MD4 named MD5 was proposed by Rivest in Both hash functions compute a bit output i e they possess a collision resis tance of about MD5 became extremely widely used e g in Internet security protocols for computing checksums of files or for storing of password hashes There were however early signs of potential weaknesses Thus the US NIST published a new message digest standard which was coined the Secure Hash Algorithm SHA in This is the first member of the SHA family and is officially called SHA even though it is nowadays commonly referred to as SHA 0 In SHA 0 was modified to SHA 1 The difference between the SHA 0 and SHA 1 algorithms lies in the schedule of the compression function to improve its cryptographic security Both algorithms have an output length of bit In a partial attack against MD5 by Hans Dobbertin led to more and more experts recommending SHA 1 as a replacement for the widely used MD5 Since then SHA 1 has gained wide adoption in numerous products and standards

In the absence of analytical attacks the maximum collision resistance of SHA 0 and SHA 1 is about which is not a good fit if they are used in protocols together with algorithms such as AES which has a security level of 6 bits Similarly most public key schemes can offer higher security levels for instance elliptic curves can have security levels of bits if bits curves are used Thus in NIST introduced three more variants of SHA 1 SHA SHA and SHA with message digest lengths of and bits respectively A further modification SHA was introduced in in order to fit the security level of 3DES These four hash functions are often referred to as SHA 2

In collision finding attacks against MD5 and SHA 0 where announced by Xiaoyun Wang One year later it was claimed that the attack could be extended to SHA 1 and it was claimed that a collision search would take steps which is considerably less than the achieved by the birthday attack Table gives an overview of the main parameters of the MD4 family

In Section we will learn about the internal functioning of SHA 1 which is to date despite its potential weakness the most widely deployed hash function

At this point we would like to note that finding a collision does not necessarily mean that the hash function is insecure in every situation There are many applica tions for hash functions e g key derivation or storage of passwords where only

The MD4 family of hash functions

preimage and second preimage resistance are required For such applications MD5 is still sufficient

Hash Functions from Block Ciphers

Hash functions can also be constructed using block cipher chaining techniques As in the case of dedicated hash functions like SHA 1 we divide the message x into blocks xi of a fixed size Figure shows a construction of such a hash function The message chunks xi are encrypted with a block cipher e of block size b As m bit key input to the cipher we use a mapping g from the previous output Hi 1 which is a b to m bit mapping In the case of b m which is for instance given if AES with a bit key is being used the function g can be the identity mapping After the encryption of the message block xi we XOR the result to the original message block The last output value computed is the hash of the whole message x1 x2 xn i e Hn h x

The Matyas Meyer Oseas hash function construction from block ciphers

The function can be expressed as

Hi eg Hi 1 xi xi

This construction which is named after its inventors is called the Matyas Meyer Oseas hash function

There exist several other variants of block cipher based realizations of hash func tions Two popular ones are shown in Figure

Davies Meyer left and Miyaguchi Preneel hash function constructions from block ciphers

The expressions for the two hash functions are

Hi Hi 1 exi Hi 1 Davies Meyer

Hi Hi 1 xi eg Hi 1 xi Miyaguchi Preneel

All three hash functions need to have initial values assigned to H0 These can be public values e g the all zero vector All schemes have in common that the bit size of the hash output is equal to the block width of the cipher used In situations where only preimage and second preimage resistance is required block ciphers like AES with bit block width can be used because they provide a security level of bit against those attacks For application which require collision resistance the bit length provided by most modern block ciphers is not sufficient The birthday attack reduces the security level to mere 64 bit which is a computational complexity that is within reach of PC clusters and certainly is doable for attackers with large budgets

One solution to this problem is to use Rijndael with a block width of or bit These bit lengths provide a security level of 96 and bit respectively against birthday attacks which is sufficient for most applications We recall from Section that Rijndael is the cipher that became AES but allows block sizes of and bit

Another way of obtaining larger message digests is to use constructions which are composed of several instances of a block cipher and which yield twice the width of the block length b Figure shows such a construction for the case that a cipher e is being employed whose key length is twice the block length This is in particular the case for AES with a bit key The message digest output are the 2b bit Hn L Hn R If AES is being used this output is 2b bit long which provides a high level of security against collision attacks As can be seen from the

figure the previous output of the left cipher Hi 1 L is fed back as input to both block

ciphers The concatenation of the previous output of the right cipher Hi 1 R with the next message block xi forms the key for both ciphers For security reasons a

constant c has to be XORed to the input of the right block cipher c can have any value other than the all zero vector As in the other three constructions described above initial values have to be assigned to the first hash values H0 L and H0 R

Hirose construction for a hash function with twice the block width

We introduce here the Hirose construction for the case that the key length be twice the block width There are many other ciphers that satisfy this condition in addition to AES e g the block ciphers Blowfish Mars RC6 and Serpent If a hash function for resource constrained applications is needed the lightweight block ci pher PRESENT cf Section allows an extremely compact hardware implemen tation With a key size of bit and a block size of 64 bit the construction com putes a bit hash output This message digest size resists preimage and second preimage attacks but offers only marginal security against birthday attacks

The Secure Hash Algorithm SHA 1

The Secure Hash Algorithm SHA 1 is the most widely used message digest func tion of the MD4 family Even though new attacks have been proposed against the algorithm it is very instructive to look at its details because the stronger versions in the SHA 2 family show a very similar internal structure SHA 1 is based on a Merkle Damga rd construction as can be seen in Figure

An interesting interpretation of the SHA 1 algorithm is that the compression function works like a block cipher where the input is the previous hash value Hi 1 and the key is formed by the message block xi As we will see below the actual

rounds of SHA 1 are in fact quite similar to a Feistel block cipher

SHA 1 produces a bit output of a message with a maximum length of bit Before the hash computation the algorithm has to preprocess the message During the actual computation the compression function processes the message in bit

High level diagram of SHA 1

chunks The compression function consists of 80 rounds which are divided into four stages of 20 rounds each

Before the actual hash computation the message x has to be padded to fit a size of a multiple of bit For the internal processing the padded message must then be divided into blocks Also the initial value H0 is set to a predefined constant

Padding Assume that we have a message x with a length of l bit To obtain an overall message size of a multiple of bits we append a single 1 followed by k zero bits and the binary 64 bit representation of l Consequently the number of required zeros k is given by

Figure illustrates the padding of a message x

0 Padding of a message in SHA 1

Example Given is the message abc consisting of three 8 bit ASCII char acters with a total length of l 24 bits

We append a 1 followed by k zero bits where k is determined by

k l 1 25 mod

Finally we append the 64 bit value which contains the binary representation of the length l 02 The padded message is then given by

Dividing the padded message Prior to applying the compression function we need to divide the message into bit blocks x1 x2 xn Each bit block can be subdivided into 16 words of size of 32 bits For instance the ith block of the message x is split into

xi x 0 x 1 x 15

where x k are words of size of 32 bits

Initial value H0 A bit buffer is used to hold the initial hash value for the first iteration The five 32 bit words are fixed and given in hexadecimal notation as

Each message block xi is processed in four stages with 20 rounds each as shown in Figure The algorithm uses

a message schedule which computes a 32 bit word W0 W1 W79 for each of the 80 rounds The words Wj are derived from the bit message block as follows

Wj 16 Wj 14 Wj 8 Wj 3 6 j 79

where X n indicates a circular left shift of the word X by n bit positions

five working registers of size of 32 bits A B C D E

a hash value Hi consisting of five 32 bit words H 0 H 1 H 2 H 3 H 4 In the

beginning the hash value holds the initial value H0 which is replaced by a new hash value after the processing of each single message block The final hash value Hn is equal to the output h x of SHA 1

1 Eighty round compression function of SHA 1

The four SHA 1 stages have a similar structure but use different internal func tions ft and constants Kt where 1 t 4 Each stage is composed of 20 rounds where parts of the message block are processed by the function ft together with

some stage dependent constant Kt The output after 80 rounds is added to the input value Hi 1 modulo in word wise fashion

The operation within round j in stage t is given by

A B C D E E ft B C D A 5 Wj Kt A B 30 C D

and is depicted in Figure The internal functions ft and constants Kt change

2 Round j in stage t of SHA 1

depending on the stage according to Table i e every 20 rounds a new function and a new constant are being used The function only uses bitwise Boolean opera tions namely logical AND OR NOT top bar and XOR These operation are applied to 32 bit variables and are very fast to implement on modern PCs

A SHA 1 round as shown in Figure has some resemblance to the round of a Feistel network Such structures are sometimes referred to as generalized Feistel networks Feistel networks are generally characterized by the fact the first part of the input is copied directly to the output The second part of the input is encrypted using the first part where the first part is sent through some function e g the f function in the case of DES In the SHA 1 round the inputs A B C and D are passed to the output with no change A C D or only minimal change rotation of B However the input word E is encrypted by adding values derived from the other four input words The message derived value Wi and the round constant play the role of subkeys

Round functions and round constants for the SHA rounds

SHA 1 was designed to be especially amenable to software implementations Each round requires only bitwise Boolean operation with 32 bit registers Somewhat countering this effect is the large number of rounds Nevertheless optimized imple mentations on modern 64 bit microprocessors can achieve throughputs of 1 Gbit sec or beyond These are highly optimized assembly code software and typical imple mentations are most likely considerably slower Generally speaking one drawback of SHA 1 and other MD4 family algorithms is that they are difficult to parallelize It is hard to execute many of the Boolean operations that constitute a round in parallel With respect to hardware SHA 1 is certainly not a truly large algorithm but there are several factors which cause it to be larger than one might expect Recent hard ware implementations on conventional FPGAs can reach a few Gbit sec which is not that groundbreaking compared to PC based implementations One reason is that the function ft depends on the stage number t Another reason is the many registers that are required to store the bit intermediate results Hence block ciphers like AES are typically smaller and faster in hardware Also in some applications hash functions built from block ciphers as described in Section are sometimes

desirable for hardware implementations

Discussion and Further Reading

MD4 family and General Remarks It is instructive to have a look at the attack history of the MD4 family A predecessor of MD4 was Rivest s MD2 hash func tion which did not appear to become widely used It is doubtful that the algorithm would withstand today s attacks The first attacks against reduced versions of MD4 the first or the last rounds were missing were developed by Boer and Bosselaers in 53 In Dobbertin showed how collisions for the full MD4 can be con structed in less than a minute on conventional PCs 61 Later Dobbertin showed that a variant of MD4 a round was not executed does not have the one wayness prop erty In Boer and Bosselaer found collisions in MD5 54 In Dobbertin was able to find collisions for the compression function of MD5 62 In order to construct a collision for the popular SHA 1 algorithm about computations have to be executed This is still a formidable task In a distributed hash collision search over the Internet was organized by Rechberger at the Technical University of Graz in Austria At the time of writing about two years into the search no collisions have been found

RIPEMD plays a somewhat special role in the MD4 family of hash func tions Unlike all SHA 1 and SHA 2 algorithms it is the only one that was not designed by NIST and NSA but rather by a team of European researchers Even though there is no indication that any of the SHA algorithms are artificially weak ened or contain backdoors introduced by the US government that is RIPEMD might appeal to some people who heavily distrust governments Currently no

successful attacks against the hash functions are known On the other hand due to its more limited deployment there has been less scrutiny by the research community with respect to RIPEMD

It is important to point out that in addition to the MD4 family numerous other al gorithms have been proposed over the years including for instance Whirlpool 12 which is related to AES Most of them did not gain widespread adoption however Entirely different from the MD4 family are hash functions which are based on al gebraic structures such as MASH 1 and MASH 2 96 Many of these algorithms were found to be insecure

SHA 3 Due to the serious attacks against SHA 1 NIST held two public workshops to assess the status of SHA and to solicit public input on its cryptographic hash function policy and standard As a consequence NIST decided to develop additional hash functions to be named SHA 3 through a public competition This approach is quite similar to the selection process of AES In the fall of 64 algorithms had been submitted to NIST At the time of writing 33 of those hash functions are still in the competition The final decision is expected in In the meantime the SHA 2 algorithm against which no attacks are known to date appears to be the safest choice when selecting a hash function

Hash Functions from Block Ciphers The four block cipher based hash functions introduced in the chapter are all provable secure This means the best possible preimage and second preimage attacks have a complexity of 2b where b is the mes sage digest length and the best possible collision attack requires 2b 2 steps The security proof only holds if the block cipher is being treated as a black box i e no possible specific weaknesses of the cipher are being exploited In addition to the four methods of building hash functions from block ciphers introduced in this chapter there are several other constructions In Problem 12 variants are treated in more detail

The Hirose construction is relatively new 92 It can also be realized with AES with a bit key and message blocks xi of length 64 bit However the efficiency is roughly half of that of the construction presented in this chapter AES with bit message blocks There are also various other methods to build hash functions with twice the output size of the block ciphers used A prominent one is MDC 2 which was originally designed for DES but works with any block cipher MDC 2 is standardized in ISO IEC

Hash functions are keyless The two most important applications of hash func tions are their use in digital signatures and in message authentication codes such as HMAC

The three security requirements for hash functions are one wayness second preimage resistance and collision resistance

Hash functions should have at least bit output length in order to withstand collision attacks bit or more is desirable for long term security

MD5 which was widely used is insecure Serious security weaknesses have been found in SHA 1 and the hash function should be phased out The SHA 2 algorithms all appear to be secure

The ongoing SHA 3 competition will result in new standardized hash functions in a few years

Compute the output of the first round of stage 1 of SHA 1 for a bit input block of

x 0 01 i e bit is one

Ignore the initial hash value H0 for this problem i e A0 B0 hex

One of the earlier applications of cryptographic hash functions was the stor age of passwords for user authentication in computer systems With this method a password is hashed after its input and is compared to the stored hashed reference password People realized early that it is sufficient to only store the hashed versions of the passwords

Assume you are a hacker and you got access to the hashed password list Of course you would like to recover the passwords from the list in order to imper sonate some of the users Discuss which of the three attacks below allow this Exactly describe the consequences of each of the attacks

Attack A You can break the one way property of h

Attack B You can find second preimages for h

Attack C You can find collisions for h

Why is this technique of storing hashed passwords often extended by the use of a so called salt A salt is a random value appended to the password before hashing Together with the hash the value of the salt is stored in the list of hashed passwords Are the attacks above affected by this technique

Is a hash function with an output length of 80 bit sufficient for this application

Draw a block digram for the following hash functions built from a block cipher

e Hi 1 xi Hi 1 xi Hi 1

e Hi 1 xi xi Hi 1

e Hi 1 xi Hi 1 xi

e xi xi Hi 1 xi Hi 1

e xi Hi 1 xi Hi 1

e xi xi Hi 1 Hi 1

e xi Hi 1 Hi 1 Hi 1

e xi Hi 1 xi Hi 1

e xi Hi 1 Hi 1 xi

We define the rate of a block cipher based hash function as follows A block cipher based hash function that processes u input bits at a time produces v output bits and performs w block cipher encryptions per input block has a rate of

What is the rate of the four block cipher constructions introduced in Section

We consider three different hash functions which produce outputs of lengths 64 and bit After how many random inputs do we have a probability of

for a collision After how many random inputs do we have a probability of

for a collision

Describe how exactly you would perform a collision search to find a pair x1 x2 such that h x1 h x2 for a given hash function h What are the memory re quirements for this type of search if the hash function has an output length of n bits

Assume the block cipher PRESENT block length 64 bits bit key is used in a Hirose hash function construction The algorithm is used to store the hashes of passwords in a computer system For each user i with password PWi the system stores

where the passwords or passphrases have an arbitrary length Within the computer system only the values yi are actually used for identifying users and giving them access

Unfortunately the password file that contains all hash values falls into your hands and you are widely known as a very dangerous hacker This in itself should not pose a serious problem as it should be impossible to recover the passwords from the hashes due to the one wayness of the hash function However you discovered a small but momentous implementation flaw in the software The constant c in the hash scheme is assigned the value c 0 Assume you also know the initial values H0 L and H0 R

What is the size of each entry yi

Assume you want to log in as user U you might be the CEO of the organization Provide a detailed description that shows that finding a value PWhack for which

takes only about steps

Which of the three general attacks against hash functions do you perform

Why is the attack not possible if c 0

In this problem we will examine why techniques that work nicely for error correction codes are not suited as cryptographic hash functions We look at a hash function that computes an 8 bit hash value by applying the following equation

Ci bi1 bi2 bi3 bi4 bi5 bi6 bi7 bi8 Every block of 8 bits constitutes an ASCII encoded character

Encode the string CRYPTO to its binary or hexadecimal representation

Calculate the 6 bit long hash value of the character string using the previously defined equation

Break the hash function by pointing out how it is possible to find meaningful character strings which result in the same hash value Provide an appropriate example

Which cruical property of hash functions is missing in this case

Message Authentication Codes MACs

A Message Authentication Code MAC also known as a cryptographic checksum or a keyed hash function is widely used in practice In terms of security function ality MACs share some properties with digital signatures since they also provide message integrity and message authentication However unlike digital signatures MACs are symmetric key schemes and they do not provide nonrepudiation One advantage of MACs is that they are much faster than digital signatures since they are based on either block ciphers or hash functions

In this chapter you will learn

The principle behind MACs

The security properties that can be achieved with MACs

How MACs can be realized with hash functions and with block ciphers

C Paar J Pelzl Understanding Cryptography

DOI 0 6 2 Qc Springer Verlag Berlin Heidelberg

Principles of Message Authentication Codes

Similar to digital signatures MACs append an authentication tag to a message The crucial difference between MACs and digital signatures is that MACs use a sym metric key k for both generating the authentication tag and verifying it A MAC is a function of the symmetric key k and the message x We will use the notation

for this in the following The principle of the MAC calculation and verification is shown in Figure

Principle of message authentication codes MACs

The motivation for using MACs is typically that Alice and Bob want to be assured that any manipulations of a message x in transit are detected For this Bob computes the MAC as a function of the message and the shared secret key k He sends both the message and the authentication tag m to Alice Upon receiving the message and m Alice verifies both Since this is a symmetric set up she simply repeats the steps that Bob conducted when sending the message She merely recomputes the authentication tag with the received message and the symmetric key

The underlying assumption of this system is that the MAC computation will yield an incorrect result if the message x was altered in transit Hence message integrity is provided as a security service Furthermore Alice is now assured that Bob was the originator of the message since only the two parties with the same secret key k have the possibility to compute the MAC If an adversary Oscar changes the message during transmission he cannot simply compute a valid MAC since he lacks the secret key Any malicious or accidental e g due to transmission errors forgery of the message will be detected by the receiver due to a failed verification of the MAC

That means from Alice s perspective Bob must have generated the MAC In terms of security services message authentication is provided

In practice a messages x is often much larger than the corresponding MAC Hence similar to hash functions the output of a MAC computation is a fixed length authentication tag which is independent of the length of the input

Together with earlier discussed characteristics of MACs we can summarize all their important properties

The last point is important to keep in mind MACs do not provide nonrepudia tion Since the two communicating parties share the same key there is no possibility to prove towards a neutral third party e g a judge whether a message and its MAC originated from Alice or Bob Thus MACs offer no protection in scenarios where either Alice or Bob is dishonest like the car buying example we described in Sec tion 1 A symmetric secret key is not tied to a certain person but rather to two parties and hence a judge cannot distinguish between Alice and Bob in case of a dispute

In practice message authentication codes are constructed in essentially two dif ferent ways from block ciphers or from hash functions In the subsequent sections of this chapter we will introduce both options for realizing MACs

MACs from Hash Functions HMAC

An option for realizing MACs is to use cryptographic hash functions such as SHA 1 as a building block One possible construction named HMAC has become very popular in practice over the last decade For instance it is used in both the Transport Layer Security TLS protocol indicated by the little lock symbol in your Web browser as well as in the IPsec protocol suite One reason for the widespread use of

the HMAC construction is that it can be proven to be secure if certain assumptions are made

The basic idea behind all hash based message authentication codes is that the key is hashed together with the message Two obvious constructions are possible The first one

is called secret prefix MAC and the second one

is known as secret suffix MAC The symbol denotes concatenation Intuitively due to the one wayness and the good scrambling properties of modern hash func tions both approaches should result in strong cryptographic checksums However as is often the case in cryptography assessing the security of a scheme can be trickier than it seems at first glance We now demonstrate weaknesses in both constructions

Attacks Against Secret Prefix MACs

We consider MACs realized as m h k x For the attack we assume that the cryptographic checksum m is computed using a hash construction as shown in Fig ure This iterated approach is used in the majority of today s hash functions The message x that Bob wants to sign is a sequence of blocks x x1 x2 xn where the block length matches the input width of the hash function Bob computes an authentication tag as

m MACK x h k x1 x2 xn

The problem is that the MAC for the message x x1 x2 xn xn 1 where xn 1 is an arbitrary additional block can be constructed from m without knowing the secret key The attack is shown in the protocol below

Note that Alice will accept the message x1 xn xn 1 as valid even though Bob only authenticated x1 xn The last block xn 1 could for instance be an appendix to an electronic contract a situation that could have serious consequences The attack is possible since the MAC of the additional message block only needs the previous hash output which is equal to Bob s m and xn 1 as input but not the

Attacks Against Secret Suffix MACs

After studying the attack above it seems to be safe to use the other basic con struction method namely m h x k However a different weakness occurs here Assume Oscar is capable of constructing a collision in the hash function i e he can find x and xO such that

The two messages x and xO can be for instance two versions of a contract which are different in some crucial aspect e g the agreed upon payment If Bob signs x with a message authentication code

m h x k m is also a valid checksum for xO i e

m h x k h xO k

The reason for this is again given by the iterative nature of the MAC computation

Whether this attack presents Oscar with an advantage depends on the parameters used in the construction As a practical example let s consider a secret suffix MAC which uses SHA 1 as hash function which has an output length of bits and a bit key One would expect that this hash offers a security level of bits

i e an attacker cannot do better than brute forcing the entire key space to forge a

message However if an attacker ex ploits the birthday paradox cf Section 3

he can forge a signature with about computations There are indications

that SHA 1 collisions can be constructed with even fewer steps so that an actual attack might be even easier In summary we conclude that the secret suffix method also does not provide the security one would like to have from a MAC construction

A hash based message authentication code which does not show the security weak ness described above is the HMAC construction proposed by Mihir Bellare Ran Canetti and Hugo Krawczyk in The scheme consists of an inner and outer hash and is visualized in Figure

The MAC computation starts with expanding the symmetric key k with zeros on the left such that the result k is b bits in length where b is the input block width of the hash function The expanded key is XORed with the inner pad which consists of the repetition of the bit pattern

MACs from Block Ciphers CBC MAC

ipad

so that a length of b bit is achieved The output of the XOR forms the first input block to the hash function The subsequent input blocks are the message blocks x1 x2 xn

The second outer hash is computed with the padded key together with the output of the first hash Here the key is again expanded with zeros and then XORed with the outer pad

opad

The result of the XOR operation forms the first input block for the outer hash The other input is the output of the inner hash After the outer hash has been computed its output is the message authentication code of x The HMAC construction can be expressed as

HMACk x h k opad h k ipad x

The hash output length l is in practice longer than the width b of an input block For instance SHA 1 has an l bit output but accepts b bit inputs It does not pose a problem that the inner hash function output does not match the input size of outer hash because hash functions have preprocessing steps to match the input string to the block width As an example Section 1 described the preprocessing for SHA 1

In terms of computational efficiency it should be noted that the message x which can be very long is only hashed once in the inner hash function The outer hash consists of merely two blocks namely the padded key and the inner hash output Thus the computational overhead introduced through the HMAC construction is very low

In addition to its computational efficiency a major advantage of the HMAC con struction is that there exists a proof of security As for all schemes which are prov able secure HMAC is not secure per se but its security is related to the security of some other building block In the case of the HMAC construction it can be shown that if an attacker Oscar can break the HMAC he can also break the hash function used in the scheme Breaking HMAC means that even though Oscar does not know the key he can construct valid authentication tags for messages Breaking the hash function means that he can either find collisions or that he can compute a hash func tion output even though he does not know the initial value IV which was the value H0 in the case of SHA 1

MACs from Block Ciphers CBC MAC

In the preceding section we saw that hash functions can be used to realize MACs An alternative method is to construct MACs from block ciphers The most popular

approach in practice is to use a block cipher such as AES in cipher block chaining CBC mode as discussed in Section 2

Figure depicts the complete setting for the application of a MAC on basis of a block cipher in CBC mode The left side shows the sender the right side the receiver This scheme is also referred to as CBC MAC

MAC built from a block cipher in CBC mode

For the generation of a MAC we have to divide the message x into blocks xi i 1 n With the secret key k and an initial value IV we can compute the first itera tion of the MAC algorithm as

where the IV can be a public but random value For subsequent message blocks we use the XOR of the block xi and the previous output yi 1 as input to the encryption algorithm

Finally the MAC of the message x x x3 xn is the output yn of the last round

In contrast to CBC encryption the values y1 y2 y3 yn 1 are not transmitted They are merely internal values which are used for computing the final MAC value m yn

As with every MAC verification involves simply repeating the operation that were used for the MAC generation For the actual verification decision we have to com

Discussion and Further Reading

pare the computed MAC mt with the received MAC value m In case mt m the message is verified as correct In case mt m the message and or the MAC value m have been altered during transmission We note that the MAC verification is dif

ferent from CBC decryption which actually reverses the encryption operation

The output length of the MAC is determined by the block size of the cipher used Historically DES was widely used e g for banking applications More recently AES is often used it yields a MAC of length bit

Galois Counter Message Authentication Code GMAC

GMAC is a variant of the Galois Counter Mode GCM introduced in Section 6 GMAC is specified in and is a mode of operation for an underlying symmet ric key block cipher In contrast to the GCM mode GMAC does not encrypt data but only computes a message authentication code GMAC is easily parallelizable which is attractive for high speed applications The use of GMAC in IPsec Encap sulating Security Payload ESP and Authentication Header AH is described in the RFC The RFC describes how to use AES in GMAC to provide data origin authentication without confidentiality within the IPsec ESP and AH GMAC can be efficiently implemented in hardware and can reach a speed of 10 Gbit sec and above

Discussion and Further Reading

Block Cipher Based MACs Historically block cipher based MACs have been the dominant method for constructing message authentication codes As early as in i e only a couple of years after the announcement of the Data Encryption Standard DES it was suggested that DES could be used to compute cryptographic checksums 39 In the following years block cipher based MACs were standard ized in the US and became popular for assuring the integrity of financial transac tions see e g the ANSI X 7 standard 3 Much more recently the NIST recom mendation 65 specifies a message authentication code algorithm based on a sym metric key block cipher CMAC which is similar to CBC MAC The AES CMAC algorithm is specified in RFC

In this chapter the CBC MAC was introduced In addition to the CBC MAC there are the OMAC and PMAC which are both constructed with block ciphers Counter with CBC MAC CCM is a mode for authenticated encryption and is de fined for use with a bit block cipher It is described in the NIST recom mendation 64 The GMAC construction is standardized in IPSec and in the NIST recommendation for Block Cipher Modes of Operation 66

Hash Function Based MACs The HMAC construction was originally proposed at the Crypto conference 14 A very accessible treatment of the scheme can be found in 15 HMAC was turned into an Internet RFC and was quickly adopted in many Internet security protocols including TLS and IPsec In both cases it protects the integrity of a message during transmission It is widely used with the hash func tions SHA 1 and MD5 and its use with RIPEMD has also been often discussed It seems likely that the switch to more modern hash functions such as SHA 2 and SHA 3 will result in more and more HMAC constructions with these hash functions

Other MAC Constructions Another type of message authentication code is based on universal hashing and is called UMAC UMAC is backed by a formal security analysis and the only internal cryptographic component is a block cipher used to generate the pseudorandom pads and internal key material The universal hash func tion is used to produce a short hash value of fixed length This hash is then XORed with a key derived pseudorandom pad The universal hash function is designed to be very fast in software e g as low as one cycle per byte on contemporary processors and is mainly based on additions of 32 bit and 64 bit numbers and multiplication of 32 bit numbers Based on the original idea by Wegman and Carter 40 numer ous schemes have been proposed e g the schemes Multilinear Modular Hashing MMH and UMAC 89 23

MACs provide two security services message integrity and message authentica tion using symmetric techniques MACs are widely used in protocols

Both of these services are also provided by digital signatures but MACs are much faster

MACs do not provide nonrepudiation

In practice MACs are either based on block ciphers or on hash functions

HMAC is a popular MAC used in many practical protocols such as TLS

As we have seen MACs can be used to authenticate messages With this prob lem we want to show the difference between two protocols one with a MAC one with a digital signature In the two protocols the sending party performs the follow ing operation

y ek1 x h k2 x

where x is the message h is a hash function such as SHA 1 e is a private key encryption algorithm denotes simple concatenation and k1 k2 are secret keys which are only known to the sender and the receiver

y ek x sigkpr h x

Provide a step by step description e g with an itemized list of what the receiver does upon receipt of y You may want to draw a block diagram for the process on the receiver s side but that s optional

For hash functions it is crucial to have a sufficiently large number of output bits with e g bits in order to thwart attacks based on the birthday paradox Why are much shorter output lengths of e g 80 bits sufficient for MACs

For your answer assume a message x that is sent in clear together with its MAC over the channel x MACk x Exactly clarify what Oscar has to do to attack this system

We study two methods for integrity protection with encryption

Assume we apply a technique for combined encryption and integrity protection in which a ciphertext c is computed as

where h is a hash function This technique is not suited for encryption with stream ciphers if the attacker knows the whole plaintext x Explain exactly how an active attacker can now replace x by an arbitrary xt of his her choosing and

compute ct such that the receiver will verify the message correctly Assume that

x and xt are of equal length Will this attack work too if the encryption is done with a one time pad

Is the attack still applicable if the checksum is computed using a keyed hash function such as a MAC

c ek1 x MACk2 x

Assume that e is a stream cipher as above

We will now discuss some issues when constructing an efficient MAC

The messages X to be authenticated consists of z independent blocks so that X x1 x2 xz where every xi consists of xi 8 bits The input blocks are consecutively put into the compression function

ci h ci 1 xi ci 1 xi

At the end the MAC value

MACk X cz k mod 28

is calculated where k is a 64 bit long shared key Describe how exactly the ef fective part of the key k can be calculated with only one known message X

Perform this attack for the following parameters and determine the key k

What is the effective key length of k

Although two different operations 28 and 28 are utilized in this MAC this MAC based signature possesses significant weaknesses To which property of the design can these be ascribed and where should one take care when con structing a cryptographic system This essential property also applies for block ciphers and hash functions

MACs are in principle also vulnerable against collision attacks We discuss the issue in the following

Assume Oscar found a collision between two messages i e

Show a simple protocol with an attack that is based on a collision

Even though the birthday paradox can still be used for constructing collisions why is it in a practical setting much harder to construct them for MACs than for hash functions Since this is the case what security is provided by a MAC with 80 bit output compared to a hash function with 80 bit output

With the cryptographic mechanisms that we have learned so far in particular sym metric and asymmetric encryption digital signatures and message authentication codes MACs one can relatively easily achieve the basic security services cf Sect 3

Confidentiality with encryption algorithms

Integrity with MACs or digital signatures

Message authentication with MACs or digital signatures

Non repudiation with digital signatures

Similarly identification can be accomplished through protocols which make use of standard cryptographic primitives

However all cryptographic mechanisms that we have introduced so far assume that keys are properly distributed between the parties involved e g between Alice and Bob The task of key establishment is in practice one of the most important and often also most difficult parts of a security system We already learned some ways of distributing keys in particular Diffie Hellman key exchange In this chapter we will learn many more methods for establishing keys between remote parties You will learn about the following important issues

How keys can be established using symmetric cryptosystems

How keys can be established using public key cryptosystems

Why public key techniques still have shortcomings for key distribution

What certificates are and how they are used

The role that public key infrastructures play

C Paar J Pelzl Understanding Cryptography

DOI 0 6 3 Qc Springer Verlag Berlin Heidelberg

In this section we introduce some terminology some thoughts on key freshness and a very basic key distribution scheme The latter is helpful for motivating the more advanced methods which will follow in this chapter

Roughly speaking key establishment deals with establishing a shared secret be tween two or more parties Methods for this can be classified into key transport and key agreement methods as shown in Fig A key transport protocol is a tech nique where one party securely transfers a secret value to others In a key agreement protocol two or more parties derive the shared secret where all parties contribute to the secret Ideally none of the parties can control what the final joint value will be

One party generates and distributes a secret key

Classification of key establishment schemes

Parties jointly generate a secret key

Key establishment itself is strongly related to identification For instance you may think of attacks by unauthorized users who join the key establishment protocol with the aim of masquerading as either Alice or Bob with the goal of establishing a secret key with the other party To prevent such attacks each party must be assured of the identity of the other entity All of these issues are addressed in this chapter

Key Freshness and Key Derivation

In many but not all security systems it is desirable to use cryptographic keys which are only valid for a limited time e g for one Internet connection Such keys are called session keys or ephemeral keys Limiting the period in which a cryptographic key is used has several advantages A major one is that there is less damage if the

key is exposed Also an attacker has less ciphertext available that was generated un der one key which can make cryptographic attacks much more difficult Moreover an attacker is forced to recover several keys if he is interested in decrypting larger parts of plaintext Real world examples where session keys are frequently gener ated include voice encryption in GSM cell phones and video encryption in pay TV satellite systems in both cases new keys are generated within a matter of minutes or sometimes even seconds

The security advantages of key freshness are fairly obvious However the ques tion now is how can key updates be realized The first approach is to simply execute the key establishment protocols shown in this chapter over and over again However as we see later there are always certain costs associated with key establishment typ ically with respect to additional communication connections and computations The latter holds especially in the case of public key algorithms which are very compu tationally intensive

The second approach to key update uses an already established joint secret key to derive fresh session keys The principal idea is to use a key derivation function KDF as shown in Fig Typically a non secret parameter r is processed to gether with the joint secret kAB between the users Alice and Bob

Principle of key derivation

An important characteristic of the key derivation function is that it should be a one way function The one way property prevents an attacker from deducing kAB should any of the session keys become compromised which in turn would allow the attacker to compute all other session keys

One possible way of realizing the key derivation function is that one party sends a nonce i e a numerical value that is used only once to the other party Both users encrypt the nonce using the shared secret key kAB by means of a symmetric cipher such as AES The corresponding protocol is shown below

An alternative to encrypting the nonce is hashing it together with kAB One way of achieving this is that both parties perform a HMAC computation with the nonce serving as the message

Rather than sending a nonce Alice and Bob can also simply encrypt a counter

cnt periodically where the ciphertext again forms the session key

or compute the HMAC of the counter

Using a counter can save Alice and Bob one communication session because unlike the case of the nonce based key derivation no value needs to be transmitted How ever this holds only if both parties know exactly when the next key derivation needs to take place Otherwise a counter synchronization message might be required

The n2 Key Distribution Problem

Until now we mainly assumed that the necessary keys for symmetric algorithms are distributed via a secure channel as depicted in the beginning of this book in Distributing keys this way is sometimes referred to as key predistribution or out of band transmission since it typically involves a different mode or band of communication e g the key is transmitted via a phone line or in a letter Even though this seems somewhat clumsy it can be a useful approach in certain practical situations especially if the number of communicating parties is not too large How ever key predistribution quickly reaches its limits even if the number of entities in a network is only moderately large This leads to the well known n2 key distribution problem

We assume a network with n users where every party is capable of communi cating with every other one in a secure fashion i e if Alice wants to communicate with Bob these two share a secret key kAB which is only known to them but not to

any of the other n 2 parties This situation is shown for the case of a network with

n 4 participants in Fig

Keys in a network with n 4 users

We can extrapolate several features of this simple scheme for the case of n users

Each user must store n 1 keys

There is a total of n n 1 n2 keys in the network

If a new user joins the network a secure channel must be established with every other user in order to upload new keys

The consequences of these observations are not very favorable if the number of users increases The first drawback is that the number of keys in the system is roughly n2 Even for moderately sized networks this number becomes quite large All these keys must be generated securely at one location which is typically some type of trusted authority The other drawback which is often more serious in prac tice is that adding one new user to the system requires updating the keys at all existing users Since each update requires a secure channel this is very burdensome

Example A mid size company with employees wants to set up secure e mail communication with symmetric keys For this purpose 7 symmetric key pairs must be generated and 9 keys must be dis tributed via secure channels Moreover if employee number joins the company all other users must receive a key update This means that secure channels to the existing employees and to the new one must be established

Obviously this approach does not work for large networks However there are many cases in practice where the number of users is i small and ii does not change frequently An example could be a company with a small number of branches which all need to communicate with each other securely Adding a new branch does not happen too often and if this happens it can be tolerated that one new key is uploaded to any of the existing branches

Key Establishment Using Symmetric Key Techniques

Symmetric ciphers can be used to establish secret session keys This is somewhat surprising because we assumed for most of the book that symmetric ciphers them selves need a secure channel for establishing their keys However it turns out that it is in many cases sufficient to have a secure channel only when a new user joins the network This is in practice often achievable for computer networks because at set up time a trusted system administrator might be needed in person anyway who can install a secret key manually In the case of embedded devices such as cell phones a secure channel is often given during manufacture i e a secret key can be loaded into the device in the factory

The protocols introduced in the following all perform key transport and not key agreement

Key Establishment with a Key Distribution Center

The protocols developed in the following rely on a Key Distribution Center KDC This is a server that is fully trusted by all users and that shares a secret key with each user This key which is named the Key Encryption Key KEK is used to securely transmit session keys to users

A necessary prerequisite is that each user U shares a unique secret key KEK kU with the key distribution center which predistributed through a secure channel Let s look what happens if one party requests a secure session from the KDC e g Alice wants to communicate with Bob The interesting part of this approach is that the KDC encrypts the session key that will eventually be used by Alice and Bob In a basic protocol the KDC generates two messages yA and yB for Alice and Bob respectively

Each message contains the session key encrypted with one of the two KEKs The protocol looks like this

The protocol begins with a request message RQST IDA IDB where IDA and IDB simply indicate the users involved in the session The actual key establishment protocol is executed subsequently in the upper part of the drawing Below the solid line is as an example shown how Alice and Bob can now communicate with each other securely using the session key

It is important to note that two types of keys are involved in the protocol The KEKs kA and kB are long term keys that do not change The session key kses is an ephemeral key that changes frequently ideally for every communication session In order to understand this protocol more intuitively one can view the predis tributed KEKs as forming a secret channel between the KDC and each user With this interpretation the protocol is straightforward The KDC simply sends a session key to Alice and Bob via the two respective secret channels

Since the KEKs are long term keys whereas the session keys have typically a much shorter lifetime in practice sometimes different encryption algorithms are used with both Let s consider the following example In a pay TV system AES might be used with the long term KEKs kU for distributing session keys kses The session keys might only have a lifetime of say one minute The session keys are used to encrypt the actual plaintext the digital TV signal in this example with a fast stream cipher A stream cipher might be required to assure real time decryption The advantage of this arrangement is that even if a session key becomes compromised only one minute s worth of multimedia data can be decrypted by an adversary Thus the cipher that is used with the session key does not necessarily need to have the same cryptographic strength as the algorithm which is used for distributing the ses sion keys On the other hand if one of the KEKs becomes compromised all prior and future traffic can be decrypted by an eavesdropper

It is easy to modify the above protocol such that we save one communication session This is shown in the following

Alice receives the session key encrypted with both KEKs kA and kB She is able to compute the session key kses from yA and can use it subsequently to encrypt the actual message she wants to send to Bob The interesting part of the protocol is that Bob receives both the encrypted message y as well as yB He needs to decrypt the latter one in order to recover the session key which is needed for computing x

Both of the KDC based protocols have the advantage that there are only n long term symmetric key pairs in the system unlike the first na ve scheme that we en countered where about n key pairs were required The n long term KEKS only need to be stored by the KDC while each user only stores his or her own KEK Most importantly if a new user Noah joins the network a secure channel only needs to be established once between the KDC and Noah to distribute the KEK kN

Even though the two protocols protect against a passive attacker i e an adversary that can only eavesdrop there are attacks if an adversary can actively manipulate messages and create faked ones

Replay Attack One weakness is that a replay attack is possible This attack makes use of the fact that neither Alice nor Bob know whether the encrypted session key they receive is actually a new one If an old one is reused key freshness is violated This can be a particularly serious issue if an old session key has become compro mised This could happen if an old key is leaked e g through a hacker or if the encryption algorithm used with an old key has become insecure due to cryptanalyt ical advances

If Oscar gets hold of a previous session key he can impersonate the KDC and resend old messages yA and yB to Alice and Bob Since Oscar knows the session key he can decipher the plaintext that will be encrypted by Alice or Bob

Key Confirmation Attack Another weakness of the above protocol is that Alice is not assured that the key material she receives from the KDC is actually for a session between her and Bob This attack assumes that Oscar is also a legitimate but malicious user By changing the session request message Oscar can trick the KDC and Alice to set up session between him and Alice as opposed to between Alice and Bob Here is the attack

The gist of the attack is that the KDC believes Alice requests a key for a session between Alice and Oscar whereas she really wants to communicate with Bob Alice assumes that the encrypted key yO is yB i e the session key encrypted under Bob s KEK kB Note that if the KDC puts a header IDO in front of yO which asso ciates it with Oscar Oscar might simply change the header to IDB In other words Alice has no way of knowing that the KDC prepared a session with her and Oscar instead she still thinks she is setting up a session with Bob Alice continues with the protocol and encrypts her actual message as y If Oscar intercepts y he can decrypt it

The underlying problem for this attack is that there is no key confirmation If key confirmation were given Alice would be assured that Bob and no other user knows the session key

A more advanced protocol that protects against both replay and key confirmation attacks is Kerberos It is in fact more than a mere key distribution protocol its main purpose is to provide user authentication in computer networks Kerberos was standardized as an RFC in and is in widespread use It is also based on

a KDC which is named the authentication sever in Kerberos terminology Let s first look at a simplified version of the protocol

Kerberos assures the timeliness of the protocol through two measures First the KDC specifies a lifetime T for the session key The lifetime is encrypted with both session keys i e it is included in yA and yB Hence both Alice and Bob are aware of the period during which they can use the session key Second Alice uses a time stamp TS through which Bob can be assured that Alice s messages are recent and are not the result of a replay attack For this Alice s and Bob s system clocks must be synchronized but not with a very high accuracy Typical values are in the range of a few minutes The usage of the lifetime parameter T and the time stamp TS prevent replay attacks by Oscar

Equally important is that Kerberos provides key confirmation and user authenti cation In the beginning Alice sends a random nonce rA to the KDC This can be considered as a challenge because she challenges the KDC to encrypt it with their

joint KEK kA If the returned challenge rAt matches the sent one Alice is assured that

the message yA was actually sent by the KDC This method to authenticate users is

known as challenge response protocol and is widely used e g for authentication of smart cards

Through the inclusion of Bob s identity IDB in yA Alice is assured that the session key is actually meant for a session between herself and Bob With the inclusion of Alice s identity IDA in both yB and yAB Bob can verify that i the KDC included a session key for a connection between him and Alice and ii that he is currently actually talking to Alice

Remaining Problems with Symmetric Key Distribution

Even though Kerberos provides strong assurance that the correct keys are being used and that users are authenticated there are still drawbacks to the protocols dis cussed so far We now describe remaining general problems that exist for KDC based schemes

Communication requirements One problem in practice is that the KDC needs to be contacted if a new secure session is to be initiated between any two parties in the network Even though this is a performance rather than a security problem it can be a serious hindrance in a system with very many users In Kerberos one can alleviate this potential problem by increasing the lifetime T of the key In practice Kerberos can run with tens of thousands of users However it would be a problem to scale such an approach to all Internet users

Secure channel during initialization As discussed earlier all KDC based proto cols require a secure channel at the time a new user joins the network for transmit ting that user s key encryption key

Single point of failure All KDC based protocols including Kerberos have the security drawback that they have a single point of failure namely the database that contains the key encryption keys the KEKs If the KDC becomes compromised all KEKs in the entire system become invalid and have to be re established using secure channels between the KDC and each user

No perfect forward secrecy If any of the KEKs becomes compromised e g through a hacker or Trojan software running on a user s computer the consequences are serious First all future communication can be decrypted by the attacker who eavesdrops For instance if Oscar got a hold of Alice s KEK kA he can recover the session key from all messages yA that the KDC sends out Even more dramatic is the fact that Oscar can also decrypt past communications if he stored old messages yA and y Even if Alice immediately realizes that her KEK has been com promised and she stops using it right away there is nothing she can do to prevent Oscar from decrypting her past communication Whether a system is vulnerable if long term keys are compromised is an important feature of a security system and there is a special terminology used

Definition A cryptographic protocol has perfect forward secrecy PFS if the compromise of long term keys does not allow an attacker to obtain past session keys

Neither Kerberos nor the simpler protocols shown earlier offer PFS The main mechanism to assure PFS is to employ public key techniques which we study in the following sections

Key Establishment Using Asymmetric Techniques

Public key algorithms are especially suited for key establishment protocols since they don t share most of the drawbacks that symmetric key approaches have In fact next to digital signatures key establishment is the other major application domain of public key schemes They can be used for both key transport and key agreement For the former Diffie Hellman key exchange elliptic curve Diffie Hellman or re lated protocols are often used For key transport any of the public key encryption schemes e g RSA or Elgamal is often used We recall at this point that public key primitives are quite slow and that for this reason actual data encryption is usually done with symmetric primitives like AES or 3DES after a key has been established using asymmetric techniques

At this moment it looks as though public key schemes solve all key establishment problems It turns out however that they all require what is termed an authenticated channel to distribute the public keys The remainder of this chapter is chiefly devoted to solving the problem of authenticated public key distribution

Man in the Middle Attack

The man in the middle attack1 is a serious attack against public key algorithms The basic idea of the attack is that the adversary Oscar replaces the public keys sent out by the participants with his own keys This is possible whenever public keys are not authenticated The man in the middle MIM attack has far reaching consequences for asymmetric cryptography For didactical reasons we will study the MIM attack against the Diffie Hellman key exchange DHKE However it is extremely important to bear in mind that the attack is applicable against any asym metric scheme unless the public keys are protected e g through certificates a topic that is discussed in Sect

We recall that the DHKE allows two parties who never met before to agree on a shared secret by exchanging messages over an insecure channel For convenience we restate the DHKE protocol here

1 The man in the middle attack should not be confused with the similarly sounding but in fact entirely different meet in the middle attack against block ciphers which was introduced in Sect 1

As we discussed in Sect if the parameters are chosen carefully which in cludes especially a prime p with a length of or more bit the DHKE is secure against eavesdropping i e passive attacks We consider now the case that an adver sary is not restricted to only listening to the channel Rather Oscar can also actively take part in the message exchange by intercepting changing and generating mes sages The underlying idea of the MIM attack is that Oscar replaces both Alice s and Bob s public key by his own The attack is shown here

Let s look at the keys that are being computed by the three players Alice Bob and Oscar The key Alice computes is

kAO B a o a oa mod p

which is identical to the key that Oscar computes as kAO Ao a o ao mod p At the same time Bob computes

kBO A b o b ob mod p

which is identical to Oscar s key kBO Bo b o bo mod p Note that the two malicious keys that Oscar sends out A and B are in fact the same values With use different names here merely to stress the fact that Alice and Bob assume that they have received each other s public keys

What happens in this attack is that two DHKEs are being performed simultane ously one between Alice and Oscar and another one between Bob and Oscar As a result Oscar has established a joined key with Alice which we termed kAO and another one with Bob which we named kBO However neither Alice nor Bob is aware of the fact that they share a key with Oscar and not with each other Both assume that they have computed a joint key kAB

From here on Oscar has much control over encrypted traffic between Alice and Bob As an example here is how he can read encrypted messages in a way that goes unnoticed by Alice and Bob

For illustrative purposes we assumed that AES is used for the encryption Of course any other symmetric cipher can be used as well Please note that Oscar can not only read the plaintext x but can also alter it prior to re encrypting it with kBO This can have serious consequences e g if the message x describes a financial transaction

The underlying problem of the man in the middle attack is that public keys are not authenticated We recall from Sect 3 that message authentication ensures that the sender of a message is authentic However in the scenario at hand Bob receives a public key which is supposedly Alice s but he has no way of knowing whether that is in fact the case To make this point clear let s examine how a key of a user Alice would look in practice

where IDA is identifying information e g Alice s IP address or her name together with date of birth The actual public key kpub A however is a mere binary string e g bit If Oscar performs a MIM attack he would change the key to

Since everything is unchanged except the anonymous actual bit string the receiver will not be able to detect that it is in fact Oscar s This observation has far reaching consequences which can be summarized in the following statement

Even though public key schemes do not require a secure channel they require authen ticated channels for the distribution of the public keys

We would like to stress here again that the MIM attack is not restricted to the DHKE but is in fact applicable to any asymmetric crypto scheme The attack always pro ceeds the same way Oscar intercepts the public key that is being sent and replaces it with his own

The problem of trusted distribution of private keys is central in modern public key cryptography There are several ways to address the problem of key authentica tion The main mechanism is the use of certificates The idea behind certificates is quite easy Since the authenticity of the message kpub A IDA is violated by an ac tive attack we apply a cryptographic mechanism that provides authentication More specifically we use digital signatures 2 Thus a certificate for a user Alice in its most basic form is the following structure

CertA kpub A IDA sigkpr kpub A IDA

The idea is that the receiver of a certificate verifies the signature prior to using the public key We recall from 0 that the signature protects the signed message

which is the structure kpub A IDA in this case against manipulation If Oscar

attempts to replace kpub A by kpub O it will be detected Thus it is said that certifi cates bind the identity of a user to their public key

Certificates require that the receiver has the correct verification key which is a public key If we were to use Alice s public key for this we would have the same problem that we are actually trying to solve Instead the signatures for certificates are provided by a mutually trusted third party This party is called the Certification Authority commonly abbreviated as CA It is the task of the CA to generate and issue certificates for all users in the system For certificate generation we can distinguish between two main cases In the first case the user computes her own asymmetric key pair and merely requests the CA to sign the public key as shown in the following simple protocol for a user named Alice

2 MACs also provide authentication and could in principle also be used for authenticating pub lic keys However because MACs themselves are symmetric algorithms we would again need a secure channel for distributing the MAC keys with all the associated drawbacks

From a security point of view the first transaction is crucial It must be assured that Alice s message kpub A IDA is sent via an authenticated channel Otherwise Oscar could request a certificate in Alice s name

In practice it is often advantageous that the CA not only signs the public keys but also generates the public private key pairs for each user In this case a basic protocol looks like this

For the first transmission an authenticated channel is needed In other words The CA must be assured that it is really Alice who is requesting a certificate and not Oscar who is requesting a certificate in Alice s name Even more sensitive is the second transmission consisting of CertA kpr A Because the private key is being sent here not only an authenticated but a secure channel is required In practice this could be a certificate delivered by mail on a CD ROM

Before we discuss CAs in more detail let s have a look at the DHKE which is protected with certificates

One very crucial point here is the verification of the certificates Obviously with out verification the signatures within the certificates would be of no use As can be seen in the protocol verification requires the public key of the CA This key must be transmitted via an authenticated channel otherwise Oscar could perform MIM attacks again It looks like we haven t gained much from the introduction of cer tificates since we again require an authenticated channel However the difference from the former situation is that we need the authenticated channel only once at set up time For instance public verification keys are nowadays often included in PC software such as Web browsers or Microsoft software products The authen ticated channel is here assumed to be given through the installation of original soft ware which has not been manipulated What s happening here from a more abstract point of view is extremely interesting namely a transfer of trust We saw in the earlier example of DHKE without certificates that Alice and Bob have to trust each other s public keys directly With the introduction of certificates they only have to trust the CA s public key kpub CA If the CA signs other public keys Alice and Bob know that they can also trust those This is called a chain of trust

Public Key Infrastructures PKI and CAs

The entire system that is formed by CAs together with the necessary support mecha nisms is called a public key infrastructure usually referred to as PKI As the reader can perhaps start to imagine setting up and running a PKI in the real world is a complex task Issues such as identifying users for certificate issuing and trusted dis tribution of CA keys have to be solved There are also many other real world issues among the most complex are the existence of many different CAs and revocation of certificates We discuss some aspects of using certificate systems in practice in the following

In practice certificates not only include the ID and the public key of a user they tend to be quite complex structures with many additional fields As an example we look at the a X certificate in Fig X is an important standard for network authentication services and the corresponding certificates are widely used for Internet communication i e in S MIME IPsec and SSL TLS

Detailed structure of an X certificate

Discussing the fields defined in a X certificate gives us some insight into many aspects of PKIs in the real world We discuss the most relevant ones in the following

Certificate Algorithm Here it is specified which signature algorithm is being used e g RSA with SHA 1 or ECDSA with SHA 2 and with which parameters e g the bit lengths

Issuer There are many companies and organizations that issue certificates This field specifies who generated the one at hand

Period of Validity In most cases a public key is not certified indefinitely but rather for a limited time e g for one or two years One reason for doing this is that private keys which belong to the certificate may become compromised By limiting the validity period there is only a certain time span during which an attacker can maliciously use the private key Another reason for a restricted lifetime is that especially for certificates for companies it can happen that the

user ceases to exist If the certificates and thus the public keys are only valid for limited time the damage can be controlled

Subject This field contains what was called IDA or IDB in our earlier examples It contains identifying information such as names of people or organizations Note that not only actual people but also entities like companies can obtain certificates

Subject s Public Key The public key that is to be protected by the certificate is here In addition to the binary string which is the public key the algorithm e g Diffie Hellman and the algorithm parameters e g the modulus p and the primitive element are stored

Signature The signature over all other fields of the certificate

We note that for every signature two public key algorithms are involved the one whose public key is protected by the certificate and the algorithm with which the certificate is signed These can be entirely different algorithms and parameter sets For instance the certificate might be signed with an RSA bit algorithm while the public key within the certificate could belong to a bit elliptic curve scheme

Chain of Certificate Authorities CAs

In an ideal world there would be one CA which issues certificates for say all In ternet users on planet Earth Unfortunately that is not the case There are many dif ferent entities that act as CAs First of all many countries have their own official CA often for certificates that are used for applications that involve government busi ness Second certificates for websites are currently issued by more than 50 mostly commercial entities Most Web browsers have the public key of those CAs pre installed Third many corporations issue certificate for their own employees and external entities who do business with them It would be virtually impossible for a user to have the private keys of all these different CAs at hand What is done instead is that CAs certify each other

Let s look at an example where Alice s certificate is issued by CA1 and Bob s by CA2 At the moment Alice is only in possession of the public key of her CA1 and Bob has only kpub CA2 If Bob sends his certificate to Alice she cannot verify Bob s public key This situation looks like this

Alice can now request CA2 s public key which is itself contained in a certificate that was signed by Alice s CA1

The structure CertCA2 contains the public key of CA2 signed by CA1 which looks like this

CertCA2 kpub CA2 IDCA2 sigkpr CA1 kpub CA2 IDCA2

The important outcome of the process is that Alice can now verify Bob s key

What s happening here is that a certificate chain is being established CA1 trusts CA2 which is expressed by CA1 signing the public key kpub CA2 Now Alice can trust Bob s public key since it was signed by CA1 This situation is called a chain of trust and it is said that trust is delegated

In practice CAs can be arranged hierarchically where each CA signes the public key of the certificate authorities one level below Alternatively CAs can cross certify each other without a strict hierarchical relationship

Certificate Revocation Lists

One major issue in practice is that it must be possible to revoke certificates A com mon reason is that a certificate is stored on a smart card which is lost Another reason could be that a person left an organization and one wants to make sure that she is not using the public key that was given to her The solution in these situations seems easy Just publish a list with all certificates that are currently invalid Such a list is called a certificate revocation list or CRL Typically the serial numbers of certificates are used to identify the revoked certificates Of course a CRL must be signed by the CA since otherwise attacks are possible

The problem with CLRs is how to transmit them to the users The most straight forward way is that every user contacts the issuing CA every time a certificate of another user is received The major drawback is that now the CA is involved in every session set up This was one major drawback of KDC based i e symmetric key approaches The promise of certificate based communication was that no online contact to a central authority was needed

An alternative is that CRLs are sent out periodically The problem with this ap proach is that there is always a period during which a certificate is invalid but users

Discussion and Further Reading

have not yet been informed For instance if the CRL is sent out at 0 am every morning a time with relatively little network traffic otherwise a dishonest person could have almost a whole day where a revoked certificate is still valid To counter this the CRL update period can be shortened say to one hour However this would be a tremendous burden on the bandwidth of the network This is an instructive ex ample for the tradeoff between costs in the form of network traffic on one hand and security on the other hand In practice a reasonable compromise must be found

In order to keep the size of CRLs moderate often only the changes from the last CRL broadcast are sent out These update only CRLs are referred to as delta CRLs

Discussion and Further Reading

Key Establishment Protocols In most modern network security protocols public key approaches are used for establishing keys In this book we introduced the Diffie Hellman key exchange and described a basic key transport protocol in cf In practice often considerably more advanced asymmetric protocols are used However most of them are based on either the Diffie Hellman or a key transport protocol A comprehensive overview on this area is given in 33 We now give a few examples of generic cryptographic protocols that are of ten preferred over the basic Diffie Hellman key exchange The MTI Matsumoto Takashima Imai protocols are an ensemble of authenticated Diffie Hellman key exchanges which were already published in Good descriptions can be found in 33 and Another popular Diffie Hellman extension is the station to station STS protocol It uses certificates and provides both user and key authentication A discussion about STS variants can be found in 60 A more recent protocol for authenticated Diffie Hellman is the MQV protocol which is discussed in It is

typically used with elliptic curves

A prominent practical example for a key establishment protocol is the Internet Key Exchange IKE protocol IKE provides key material for IPsec which is the official security mechanism for Internet traffic IKE is quite complex and offers many options At its core however is a Diffie Hellman key agreement followed by an authentication The latter can either be achieved with certificates or with pre shared keys A good starting point for more information on IPsec and IKE is the RFC and more accessibly reference Chapter 16

Certificates and Alternatives During the second half of the s there was a belief that essentially every Internet user would need a certificate in order to com municate securely e g for doing ebusiness transactions PKI was a buzzword for some time and many companies were formed that provided certificates and PKI ser vices However it turned out that there are major technical and practical hurdles to a PKI that truly encompasses all or most Internet users What has happened instead is that nowadays many servers are authenticated with certificates for instance Internet retailers whereas most individual users are not The needed CA verification keys

are often preinstalled in users Web browsers This asymmetric set up the server is authenticated but the user is not is acceptable since the user is typically the one who provides crucial information such as her credit card number A comprehensive introduction to the large field of PKI and certificates is given in the book 2 An in teresting and entertaining discussion about the alleged shortcomings of PKI is given in 74 and an equally instructive rebuttal is online at

We introduced certificates and a public key infrastructure as the main method for authenticating public keys Such hierarchical organized certificates are only one possible approach though this is the most widely used one Another concept is the web of trust that relies entirely on trust relationships between parties The idea is as follows If Alice trusts Bob it is assumed that she also wants to trust all other users whom Bob trusts This means that every party in such a web of trust implicitly trusts parties whom it does not know or has never met before The most popular example for such a system are Pretty Good Privacy PGP and Gnu Privacy Guard GPG which are widely used for signing and encrypting emails

A key transport protocol securely transfers a secret key to other parties

In a key agreement protocol two or more parties negotiate a common secret key

In most common symmetric protocols the key exchange is coordinated by a trusted third party A secure channel between the third party and each user is only required at set up time

Symmetric key establishment protocols do not scale well to networks with large numbers of users and they provide typically no perfect forward secrecy

The most widely used asymmetric key establishment protocol is the Diffie Hellman key exchange

All asymmetric protocols require that the public keys are authenticated e g with certificates Otherwise man in the middle attacks are possible

In this exercise we want to analyze some variants of key derivation In prac tice one masterkey kMK is exchanged in a secure way e g certificate based DHKE between the involved parties Afterwards the session keys are regularly updated by use of key derivation For this purpose three different methods are at our disposal

k0 kMK ki 1 ki 1

k0 h kMK ki 1 h ki

k0 h kMK ki 1 h kMK i ki

where h marks a secure hash function and ki is the ith session key

What are the main differences between these three methods

Which method provides Perfect Forward Secrecy

Assume Oscar obtains the nth session key e g via brute force Which sessions can he now decrypt depending on the chosen method

Which method remains secure if the masterkey kMK is compromised Give a rationale

Imagine a peer to peer network where users want to communicate in an authenticated and confidential way without a central Trusted Third Party TTP

How many keys are collectively needed if symmetric algorithms are deployed

How are these numbers changed if we bring in a central instance Key Distribu tion Center KDC

What is the main advantage of a KDC against the scenario without a KDC

How many keys are necessary if we make use of asymmetric algorithms

Also differentiate between keys which every user has to store and keys which are collectively necessary

You have to choose the cryptographic algorithms for a KDC where two differ ent classes of encryption occur

ekU KDC where U denotes an arbitrary network node user

ekses for the communication between two users

You have the choice between two different algorithms DES and 3DES Triple DES and you are advised to use distinct algorithms for both encryption classes Which algorithm do you use for which class Justify your answer including aspects of security as well as celerity

This exercise considers the security of key establishment with the aid of a KDC Assume that a hacker performs a successful attack against the KDC at the point of time tx where all keys are compromised The attack is detected

Which practical measures have to be taken in order to prevent decryption of future communication between the network nodes

Which steps did the attacker have to take in order to decipher data transmissions which occurred at an earlier time t tx Does such a KDC system provide Perfect Forward Secrecy PFS or not

We will now analyze an improved KDC system In contrast to the previous problem all keys ekU KDC are now refreshed in relatively short intervals

The KDC generates a new random key k i 1

The KDC transmits the new key to user U encrypted with the old one

Which decryptions are possible if a staff member of the KDC is corruptible and

sells all recent keys e i

of the KDC at the point of time tx We assume that

this circumstance is not detected until the point of time ty which could be much later e g one year

Show a key confirmation attack against the basic KDC protocol introduced in Sect Describe each step of the attack Your drawing should look similar to the one showing a key confirmation attack against the second modified KDC based protocol

Show that PFS is in fact not given in the simplified Kerberos protocol Show how Oscar can decrypt past and future communications if

Alice s KEK kA becomes compromised

Bob s KEK kB becomes compromised

Extend the Kerberos protocol such that a mutual authentication between Alice and Bob is performed Give a rationale that your solution is secure

People at your new job are deeply impressed that you worked through this book As the first job assignment you are asked to design a digital pay TV system which uses encryption to prevent service theft through wire tapping As key ex change protocol a strong Diffie Hellman with e g bit modulus is being used However since your company wants to use cheap legacy hardware only DES is available for data encryption algorithm You decide to use the following key deriva tion approach

K i f KAB II i

where f is an irreversible function

First we have to determine whether the attacker can store an entire movie with reasonable effort in particular cost Assume the data rate for the TV link is 1 Mbit s and that the longest movies we want to protect are 2 hours long How many Gbytes where 1M and 1G of data must be stored for a 2 hour film don t mix up bit and byte here Is this realistic

We assume that an attacker will be able to find a DES key in 10 minutes using a brute force attack Note that this is a somewhat optimistic assumption from an attacker s point of view but we want to provide some medium term security by assuming increasingly faster key searches in the future

How frequently must a key be derived if the goal is to prevent an offline decryp tion of a 2 hour movie in less than 30 days

We consider a system in which a key kAB is established using the Diffie Hellman key exchange protocol and the encryption keys k i are then derived by computing

k i h kAB II i

where i is just an integer counter represented as a 32 bit variable The values of i are public e g the encrypting party always indicates which value for i was used in a header that precedes each ciphertext block The derived keys are used for the actual data encryption with a symmetric algorithm New keys are derived every 60 sec during the communication session

Assume the Diffie Hellman key exchange is done with a bit prime and the encryption algorithm is AES Why doesn t it make cryptographic sense to use the key derivation protocol described above Describe the attack that would require the least computational effort from Oscar

Assume now that the Diffie Hellman key exchange is done with a bit prime and the encryption algorithm is DES Describe in detail what the advan tages are that the key derivation scheme offers compared to a system that just uses the Diffie Hellman key for DES

We reconsider the Diffie Hellman key exchange protocol Assume now that Oscar runs an active man in the middle attack against the key exchange as explained in Sect For the Diffie Hellman key exchange use the parameters p

2 and a b 57 for Alice and Bob respectively Oscar uses the value

o 16 Compute the key pairs kAO and kBO i the way Oscar computes them and

ii the way Alice and Bob compute them

We consider the Diffie Hellman key exchange scheme with certificates We have a system with the three users Alice Bob and Charley The Diffie Hellman algorithm uses p 61 and 18 The three secret keys are a 11 b 22 and c 33 The three IDs are ID A 1 ID B 2 and ID C 3

For signature generation the Elgamal signature scheme is used We apply the system parameters pt dt t 2 and The CA uses the ephemeral keys kE and for Alice s Bob s and Charley s signatures respec

tively In practice the CA should use a better pseudorandom generator to obtain the kE values

To obtain the certificates the CA computes xi 4 bi ID i and uses this value as input for the signature algorithm Given xi ID i follows then from ID i

Compute three certificates CertA CertB and CertC

Verify all three certificates

Compute the three session keys kAB kAC and kBC

Assume Oscar attempts to use an active substitution attack against the Diffie Hellman key exchange with certificates in the following ways

Alice wants to communicate with Bob When Alice obtains C B from Bob Os car replaces it with a valid C O How will this forgery be detected

Same scenario Oscar tries now to replace only Bob s public key bB with his own public key bO How will this forgery be detected

We consider certificate generation with CA generated keys Assume the sec ond transmission of CertA kpr A takes place over an authenticated but insecure channel i e Oscar can read this message

Show how he can decrypt traffic which is encrypted by means of a Diffie Hellman key that Alice and Bob generated

Can he also impersonate Alice such that he computes a DH key with Bob without Bob noticing

Given is a user domain in which users share the Diffie Hellman parame ters and p Each user s public Diffie Hellman key is certified by a CA Users communicate securely by performing a Diffie Hellman key exchange and then en crypting decrypting messages with a symmetric algorithm such as AES

Assume Oscar gets hold of the CA s signature algorithm and especially its pri vate key which was used to generate certificates Can he now decrypt old cipher texts which were exchanged between two users before the CA signature algorithm was compromised and which Oscar had stored Explain your answer

Another problem in certificate systems is the authenticated distribution of the CA s public key which is needed for certificate verification Assume Oscar has full control over all of Bob s communications that is he can alter all messages to and from Bob Oscar now replaces the CA s public key with his own note that Bob has no means to authenticate the key that he receives so he thinks that he received the CA public key

Certificate issuing Bob requests a certificate by sending a request containing

1 Bob s ID ID B and 2 Bob s public key B from the CA Describe exactly what Oscar has to do so that Bob doesn t find out that he has the wrong public CA key

Protocol execution Describe what Oscar has to do to establish a session key with Bob using the authenticated Diffie Hellman key exchange such that Bob thinks he is executing the protocol with Alice

Draw a diagram that shows a key transport protocol shown in from Sect in which RSA encryption is used

We consider RSA encryption with certificates in which Bob has the RSA keys Oscar manages to send Alice a verification key kpr CA which is in fact Oscar s key Show an active attack in which he can decipher encrypted messages that Alice sends to Bob Should Oscar run a MIM attack or should he set up a session only between himself and Alice

Pretty Good Privacy PGP is a widespread scheme for electronic mail se curity to provide authentication and confidentiality PGP does not necessarily re quire the use of certificate authorities Describe the trust model of PGP and how the public key management works in practice

Michel Abdalla Mihir Bellare and Phillip Rogaway DHAES An encryption scheme based on the Diffie Hellman problem Available at citeseer ist psu edu abdalla99dhaes html

Carlisle Adams and Steve Lloyd Understanding PKI Concepts Standards and Deployment

Considerations Addison Wesley Longman Publishing Boston MA USA

ANSI X9 American National Standard X 7 Financial Institution Key Manage ment

ANSI X9 American National Standard X 1 Appendix A Public Key Cryp tography Using Reversible Algorithms for the Financial Services Industry rDSA Technical report Accredited Standards Committee X9 Available at

ANSI X9 Public Key Cryptography for the Financial Services Industry Agreement

of Symmetric Keys Using Discrete Logarithm Cryptography Technical report American Bankers Association

ANSI X9 The Elliptic Curve Digital Signature Algorithm ECDSA Technical report American Bankers Association

ANSI X9 Elliptic Curve Key Agreement and Key Transport Protocols Technical report American Bankers Association

Frederik Armknecht Algebraic attacks on certain stream ciphers PhD thesis Department of Mathematics University of Mannheim Germany December

Standards for Efficient Cryptography SEC 1 Elliptic Curve Cryptography September

Daniel V Bailey and Christof Paar Efficient arithmetic in finite field extensions with appli cation in elliptic curve cryptography Journal of Cryptology 14

Elad Barkan Eli Biham and Nathan Keller Instant Ciphertext Only Cryptanalysis of GSM Encrypted Communication Journal of Cryptology 9

P S L M Barreto and V Rijmen The whirlpool hashing function September revised May paulobarreto WhirlpoolPage html

F L Bauer Decrypted Secrets Methods and Maxims of Cryptology Springer 4th edition

Mihir Bellare Ran Canetti and Hugo Krawczyk Keying Hash Functions for Message Au thentication In CRYPTO 96 Proceedings of the 16th Annual International Cryptology Conference Advances in Cryptology pages 5 Springer

Mihir Bellare Ran Canetti and Hugo Krawczyk Message Authentication using Hash Functions The HMAC Construction CRYPTOBYTES 2

C H Bennett E Bernstein G Brassard and U Vazirani The strengths and weaknesses of quantum computation SIAM Journal on Computing 23

C Paar J Pelzl Understanding Cryptography

DOI 0 6 3 Qc Springer Verlag Berlin Heidelberg

Daniel J Bernstein Multidigit multiplication for mathematicians URL http cr yp to papers html

Daniel J Bernstein Johannes Buchmann and Erik Dahmen Post Quantum Cryptography Springer

N Biggs Discrete Mathematics Oxford University Press New York 2nd edition

E Biham A fast new DES implementation in software In Fourth International Workshop on Fast Software Encryption volume of LNCS pages 2 Springer

Eli Biham and Adi Shamir Differential Cryptanalysis of the Data Encryption Standard Springer

Alex Biryukov Adi Shamir and David Wagner Real time cryptanalysis of A on a PC In

FSE Fast Software Encryption pages 8 Springer

J Black S Halevi H Krawczyk T Krovetz and P Rogaway UMAC Fast and secure message authentication In CRYPTO 99 Proceedings of the 19th Annual International Cryptology Conference Advances in Cryptology volume 99 pages 3 Springer

I Blake G Seroussi N Smart and J W S Cassels Advances in Elliptic Curve Cryptogra phy London Mathematical Society Lecture Note Series Cambridge University Press New York NY USA

Ian F Blake G Seroussi and N P Smart Elliptic Curves in Cryptography Cambridge University Press New York NY USA

Daniel Bleichenbacher Wieb Bosma and Arjen K Lenstra Some remarks on Lucas based cryptosystems In CRYPTO 95 Proceedings of the 15th Annual International Cryptology Conference Advances in Cryptology pages 6 Springer

L Blum M Blum and M Shub A simple unpredictable pseudorandom number generator

SIAM J Comput 3

Manuel Blum and Shafi Goldwasser An efficient probabilistic public key encryption scheme which hides all partial information In CRYPTO 84 Proceedings of the 4th Annual Interna tional Cryptology Conference Advances in Cryptology pages 2

Andrey Bogdanov Gregor Leander Lars R Knudsen Christof Paar Axel Poschmann Matthew J B Robshaw Yannick Seurin and Charlotte Vikkelsoe PRESENT An Ultra Lightweight Block Cipher In CHES 07 Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems number in LNCS pages 6 Springer

Dan Boneh and Matthew Franklin Identity based encryption from the Weil pairing SIAM

J Comput 5

Dan Boneh and Richard J Lipton Algorithms for black box fields and their application to cryptography extended abstract In CRYPTO 96 Proceedings of the 16th Annual Interna tional Cryptology Conference Advances in Cryptology pages 7 Springer

Dan Boneh Ron Rivest Adi Shamir and Len Adleman Twenty Years of Attacks on the RSA Cryptosystem Notices of the AMS 13

Colin A Boyd and Anish Mathuria Protocols for Key Establishment and Authentication Springer

ECC Brainpool ECC Brainpool Standard Curves and Curve Generation http

Johannes Buchmann and Jintai Ding editors Post Quantum Cryptography Second Interna tional Workshop PQCrypto Proceedings volume of LNCS Springer

Johannes Buchmann and Jintai Ding editors PQCrypto International Workshop on Post Quantum Cryptography LNCS Springer

German Federal Office for Information Security BSI http www bsi de english publications bsi standards index htm

Mike Burmester and Yvo Desmedt A secure and efficient conference key distribution system extended abstract In Advances in Cryptology EUROCRYPT 94 pages 6

C M Campbell Design and specification of cryptographic capabilities NBS Special Pub lication Computer Security and the Data Encryption Standard U S Department of Commerce National Bureau of Standards pages 6

J L Carter and M N Wegman New hash functions and their use in authentication and set equality Journal of Computer and System Sciences 7

C etin Kaya Koc Tolga Acar and Burton S Kaliski Analyzing and comparing Montgomery

multiplication algorithms IEEE Micro 3

P Chodowiec and K Gaj Very compact FPGA implementation of the AES algorithm In

C D Walter C K Koc and C Paar editors CHES 03 Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems volume of LNCS pages 3 Springer

C Cid S Murphy and M Robshaw Algebraic Aspects of the Advanced Encryption Stan

H Cohen G Frey and R Avanzi Handbook of Elliptic and Hyperelliptic Curve Cryp tography Discrete Mathematics and Its Applications Chapman and Hall CRC September

T Collins D Hopkins S Langford and M Sabin Public key cryptographic apparatus and

method United States Patent US 59 Jan

Common Criteria for Information Technology Security Evaluation commoncriteriaportal org

COPACOBANA A Cost Optimized Parallel Code Breaker

Sony Corporation Clefia new block cipher algorithm based on state of

the art design technologies Press 28E index html

Ronald Cramer and Victor Shoup A practical public key cryptosystem provably secure

against adaptive chosen ciphertext attack CRYPTO 98 Proceedings of the 18th Annual International Cryptology Conference Advances in Cryptology 5

Cryptool Educational Tool for Cryptography and Cryptanalysis

J Daemen and V Rijmen AES Proposal Rijndael In First Advanced Encryption Standard

AES Conference Ventura California USA

Joan Daemen and Vincent Rijmen The Design of Rijndael Springer

B den Boer and A Bosselaers An attack on the last two rounds of MD4 In CRYPTO 91 Proceedings of the 11th Annual International Cryptology Conference Advances in Cryptol ogy LNCS pages 3 Springer

B den Boer and A Bosselaers Collisions for the compression function of MD5 In Advances

in Cryptology EUROCRYPT 93 LNCS pages 4 Springer

Alexander W Dent A brief history of provably secure public key encryption Cryptology ePrint Archive Report

Diehard Battery of Tests of Randomness CD

W Diffie The first ten years of public key cryptography Innovations in Internetworking

W Diffie and M E Hellman New directions in cryptography IEEE Transactions on Infor mation Theory IT 54

W Diffie and M E Hellman Exhaustive cryptanalysis of the NBS Data Encryption Stan

dard COMPUTER 4 June

Whitfield Diffie Paul C Van Oorschot and Michael J Wiener Authentication and authenti cated key exchanges Des Codes Cryptography 5

Hans Dobbertin Alf swindles Ann CRYPTOBYTES

Hans Dobbertin The status of MD5 after a recent attack CRYPTOBYTES

Saar Drimer Tim Gu neysu and Christof Paar DSPs BRAMs and a Pinch of Logic New Recipes for AES on FPGAs IEEE Symposium on Field Programmable Custom Computing Machines FCCM 0 08

Morris Dworkin Recommendation for Block Cipher Modes of Opera

tion The CCM Mode for Authentication and Confidentiality May updated July pdf

Morris Dworkin Recommendation for Block Cipher Modes of Operation The CMAC Mode for Authentication NIST Special Publication D May nist gov publications nistpubs D SP D pdf

Morris Dworkin Recommendation for Block Cipher Modes of Operation Ga lois Counter Mode GCM and GMAC NIST Special Publication D November D SP D pdf

H Eberle and C P Thacker A 1 GBIT second GaAs DES chip In Custom Integrated Circuits

Conference pages IEEE

eSTREAM The ECRYPT Stream Cipher Project http www ecrypt eu org stream

The Side Channel Cryptanalysis Lounge uni bochum de en sclounge html

Thomas Eisenbarth Sandeep Kumar Christof Paar Axel Poschmann and Leif Uhsadel A Survey of Lightweight Cryptography Implementations IEEE Design Test of Computers Special Issue on Secure ICs for Secure Embedded Computing Novem ber December

S E Eldridge and C D Walter Hardware implementation of Montgomery s modular mul tiplication algorithm IEEE Transactions on Computers 9 July

T ElGamal A public key cryptosystem and a signature scheme based on discrete logarithms

IEEE Transactions on Information Theory IT 2

C Ellison and B Schneier Ten risks of PKI What you re not being told about public key infrastructure Computer Security Journal See also

M Feldhofer J Wolkerstorfer and V Rijmen AES implementation on a grain of sand

Information Security IEE Proceedings 1 0

Amos Fiat and Adi Shamir How to prove yourself practical solutions to identification and signature problems In CRYPTO 86 Proceedings of the 6th Annual International Cryptol ogy Conference Advances in Cryptology pages 4 Springer

Federal Information Processing Standards Publications FIPS PUBS

http www itl nist gov fipspubs index htm

Electronic Frontier Foundation Frequently Asked Questions FAQ About the Electronic Frontier Foundation s DES Cracker Machine eff des faq html

J Franke T Kleinjung C Paar J Pelzl C Priplata and C Stahlke SHARK A Realizable

Special Hardware Sieving Device for Factoring bit Integers In Josyula R Rao and Berk Sunar editors CHES 05 Proceedings of the 7th International Workshop on Cryptographic Hardware and Embedded Systems volume of LNCS pages 0 Springer August

Bundesamt fu r Sicherheit in der Informationstechnik Anwendungshinweise und Interpretationen zum Schema AIS Funktionalita tsklassen und Evaluations methodologie fu r physikalische Zufallszahlengeneratoren AIS 31 Version 1

Oded Goldreich Foundations of Cryptography Basic Tools Cambridge University Press

New York NY USA

Oded Goldreich Zero Knowledge A tutorial by Oded Goldreich http

Oded Goldreich Foundations of Cryptography Volume 2 Basic Applications Cambridge

University Press New York NY USA

Oded Goldreich On post modern cryptography Cryptology ePrint Archive Report

Jovan Dj Golic On the security of shift register based keystream generators In Fast Software Encryption Cambridge Security Workshop pages 00 Springer

Tim Good and Mohammed Benaissa AES on FPGA from the fastest to the smallest CHES 05 Proceedings of the 7th International Workshop on Cryptographic Hardware and Em bedded Systems pages 0

L Grover A fast quantum mechanical algorithm for database search In Proceedings of the Twenty eighth Annual ACM Symposium on Theory of Computing pages 9 ACM

Tim Gu neysu Timo Kasper Martin Novotny Christof Paar and Andy Rupp Cryptanalysis with COPACOBANA IEEE Transactions on Computers 1

S Halevi and H Krawczyk MMH message authentication in software in the Gbit second rates In Proceedings of the 4th Workshop on Fast Software Encryption volume pages 9 Springer

D R Hankerson A J Menezes and S A Vanstone Guide to Elliptic Curve Cryptography Springer

M Hellman A cryptanalytic time memory tradeoff IEEE Transactions on Information Theory 6

Shoichi Hirose Some plausible constructions of double block length hash functions In

FSE Fast Software Encryption volume of LNCS pages 5 Springer

Deukjo Hong Jaechul Sung and Seokhie Hong et al Hight A new block cipher suitable for low resource device In CHES 06 Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems pages 9 Springer

International Organization for Standardization ISO ISO IEC 8 6

International Organization for Standardization ISO ISO IEC 1 2

International Organization for Standardization ISO ISO IEC Information technology Security techniques Hash functions Part 4 Hash functions using modular arithmetic

D Kahn The Codebreakers The Story of Secret Writing Macmillan

Jens Peter Kaps Gunnar Gaubatz and Berk Sunar Cryptography on a speck of dust Com puter 4

A Karatsuba and Y Ofman Multiplication of multidigit numbers on automata Soviet Physics Doklady English translation 6

Ann Hibner Koblitz Neal Koblitz and Alfred Menezes Elliptic curve cryptography The serpentine course of a paradigm shift Cryptology ePrint Archive Report

Neal Koblitz Introduction to Elliptic Curves and Modular Forms Springer

Neal Koblitz The uneasy relationship between mathematics and cryptography Notices of the AMS pages 9 September

Neal Koblitz Alfred Menezes and Scott Vanstone The state of elliptic curve cryptography

Des Codes Cryptography 3 3

C etin Kaya Koc Cryptographic Engineering Springer

S Kumar C Paar J Pelzl G Pfeiffer and M Schimmler Breaking ciphers with COPACOBANA A cost optimized parallel code breaker In CHES 06 Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems LNCS Springer October

Matthew Kwan Reducing the Gate Count of Bitslice DES http

Ben Laurie Seven and a Half Non risks of PKI What You Shouldn t Be Told about Public Key Infrastructure

Laurie Law Alfred Menezes Minghua Qu Jerry Solinas and Scott Vanstone An efficient

protocol for authenticated key agreement Des Codes Cryptography 4

Arjen K Lenstra and Eric R Verheul The XTR public key system In CRYPTO 00 Pro ceedings of the 20th Annual International Cryptology Conference Advances in Cryptology pages 9 Springer

Rudolf Lidl and Harald Niederreiter Introduction to Finite Fields and Their Applications Cambridge University Press 2nd edition

Chae Hoon Lim and Tymur Korkishko mCrypton A lightweight block cipher for security of low cost RFID tags and Sensors In Information Security Applications volume pages 8 Springer

Yehuda Lindell Composition of Secure Multi Party Protocols A Comprehensive Study Springer

Stefan Mangard Elisabeth Oswald and Thomas Popp Power Analysis Attacks Revealing the Secrets of Smart Cards Advances in Information Security Springer

Mitsuru Matsui Linear cryptanalysis method for DES cipher In Advances in Cryptology EUROCRYPT 93

Mitsuru Matsui How far can we go on the x64 processors In FSE Fast Software Encryp tion volume of LNCS pages 8 Springer

Mitsuru Matsui and S Fukuda How to maximize software performance of symmetric prim itives on Pentium III and 4 processors In FSE Fast Software Encryption volume of LNCS pages 2 Springer

Mitsuru Matsui and Junko Nakajima On the power of bitslice implementation on Intel Core2 processor In CHES 07 Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems pages 4 Springer

Ueli M Maurer and Stefan Wolf The relationship between breaking the Diffie Hellman protocol and computing discrete logarithms SIAM Journal on Computing

D McGrew and J Viega RFC The Use of Galois Message Authentication Code GMAC in IPsec ESP and AH Technical report Corporation for National Research Initia tives Internet Engineering Task Force Network Working Group May Available at

A J Menezes P C van Oorschot and S A Vanstone Handbook of Applied Cryptography

CRC Press Boca Raton Florida USA

Ralph C Merkle Secure communications over insecure channels Commun ACM 9

Sean Murphy and Matthew J B Robshaw Essential algebraic structure within the AES In CRYPTO 02 Proceedings of the 22nd Annual International Cryptology Conference Ad vances in Cryptology pages 6 Springer

David Naccache and David M Rahi Cryptographic smart cards IEEE Micro 4

Block Cipher Modes Workshops BCM workshops html

NIST test suite for random numbers

National Institute of Standards and Technology NIST Digital Signature Stan dards DSS FIPS1 Technical report Federal Information Processing Stan dards Publication FIPS June Available at publications fips fips1 fips 1 pdf

J Nechvatal Public key cryptography In Gustavus J Simmons editor Contemporary

Cryptology The Science of Information Integrity pages 8 IEEE Press Piscataway NJ USA

Security Architecture for the Internet Protocol http www rfc editor org rfc rfc txt

I Niven H S Zuckerman and H L Montgomery An Introduction to the Theory of Numbers 5th Edition Wiley

NSA Suite B Cryptography cryptography index shtml

Philippe Oechslin Making a Faster Cryptanalytic Time Memory Trade Off In CRYPTO 03 Proceedings of the 23rd Annual International Cryptology Conference Advances in Cryptology volume of LNCS pages 0

The OpenSSL Project

European Parliament Directive 93 EC of the European Parliament and of the Council of 13 December on a Community framework for electronic signatures l en pdf

D Pointcheval and J Stern Security proofs for signature schemes In U Maurer editor Ad

vances in Cryptology EUROCRYPT 96 volume of LNCS pages 8 Springer

Axel Poschmann Lightweight Cryptography Cryptographic Engineering for a Pervasive World PhD thesis Department of Electrical Engineering and Com puter Sciences Ruhr University Bochum Germany April crypto ruhr uni bochum de en theses html

B Preneel R Govaerts and J Vandewalle Hash functions based on block ciphers A syn

thetic approach LNCS 7 78

Bart Preneel MDC 2 and MDC 4 In Henk C A van Tilborg editor Encyclopedia of Cryptography and Security Springer

Electronic Signatures in Global and National Commerce Act United States of America

Jean Jacques Quisquater Louis Guillou Marie Annick and Tom Berson How to explain zero knowledge protocols to your children In CRYPTO 89 Proceedings of the 9th Annual International Cryptology Conference Advances in Cryptology pages 1 Springer

M O Rabin Digitalized Signatures and Public Key Functions as Intractable as Factoriza tion Technical report Massachusetts Institute of Technology

W Rankl and W Effing Smart Card Handbook John Wiley Sons Inc

R L Rivest A Shamir and L Adleman A method for obtaining digital signatures and

public key cryptosystems Communications of the ACM 6 February

Ron Rivest The RC4 Encryption Algorithm March

Dorothy Elizabeth Robling Denning Cryptography and Data Security Addison Wesley Longman Publishing Co Inc

Matthew Robshaw and Olivier Billet editors New Stream Cipher Designs The eSTREAM Finalists volume of LNCS Springer

Carsten Rolfes Axel Poschmann Gregor Leander and Christof Paar Ultra lightweight im plementations for smart devices security for gate equivalents In Proceedings of the 8th Smart Card Research and Advanced Application IFIP Conference CARDIS volume of LNCS pages 03 Springer

K H Rosen Elementary Number Theory 5th Edition Addison Wesley

Public Key Cryptography Standard PKCS http www rsasecurity com rsalabs node asp id

Claus Peter Schnorr Efficient signature generation by smartcards Journal of Cryptology 74

A Shamir Factoring large numbers with the TWINKLE device In CHES 99 Proceed ings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems volume of LNCS pages 2 Springer August

A Shamir and E Tromer Factoring Large Numbers with the TWIRL Device In CRYPTO 03 Proceedings of the 23rd Annual International Cryptology Conference Advances in Cryptology volume of LNCS pages 6 Springer

P Shor Polynomial time algorithms for prime factorization and discrete logarithms SIAM Journal on Computing Communication Theory of Secrecy Systems 09

J H Silverman The Arithmetic of Elliptic Curves Springer

J H Silverman Advanced Topics in the Arithmetic of Elliptic Curves Springer

J H Silverman A Friendly Introduction to Number Theory Prentice Hall 3rd edition

Simon Singh The Code Book The Science of Secrecy from Ancient Egypt to Quantum Cryptography Anchor August

Jerome A Solinas Efficient arithmetic on Koblitz curves Designs Codes and Cryptography 3 9

J H Song R Poovendran J Lee and T Iwata RFC The AES CMAC Algorithm Technical report Corporation for National Research Initiatives Inter net Engineering Task Force Network Working Group June Available at

NIST Special Publication SP D Recommendation for Block Cipher Modes of Oper

ation Galois Counter Mode GCM and GMAC November Available at http csrc nist gov publications nistpubs D SP D pdf

W Stallings Cryptography and Network Security Principles and Practice Prentice Hall 4th edition

Tsuyoshi Takagi Fast RSA type cryptosystem modulo pkq In CRYPTO 98 Proceedings of the 18th Annual International Cryptology Conference Advances in Cryptology pages

S Trimberger R Pang and A Singh A 12 Gbps DES Encryptor Decryptor Core in an FPGA In C K Koc and C Paar editors CHES 00 Proceedings of the 2nd International Workshop on Cryptographic Hardware and Embedded Systems volume of LNCS pages 3 Springer August 8

Trivium Specifications trivium trivium p3 pdf

Walter Tuchman A brief history of the data encryption standard In Internet Besieged Countering Cyberspace Scofflaws pages 0 ACM Press Addison Wesley

Annual Workshop on Elliptic Curve Cryptography ECC uwaterloo ca conferences

Digital Signature Law Survey

Henk C A van Tilborg editor Encyclopedia of Cryptography and Security Springer

Ingrid Verbauwhede Frank Hoornaert Joos Vandewalle and Hugo De Man ASIC cryptographical processor based on DES ucla edu pdf euroasic pdf

SHARCS Special purpose Hardware for Attacking Cryptographic Systems

WAIFI International Workshop on the Arithmetic of Finite Fields http

Andre Weimerskirch and Christof Paar Generalizations of the Karatsuba algo rithm for efficient implementations Cryptology ePrint Archive Report

D Whiting R Housley and N Ferguson RFC Counter with CBC MAC CCM

Technical report Corporation for National Research Initiatives Internet Engineering Task Force Network Working Group September

M J Wiener Efficient DES Key Search An Update CRYPTOBYTES Autumn

Thomas Wollinger Jan Pelzl and Christof Paar Cantor versus Harley Optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems IEEE Transactions on Computers 2

Advanced Encryption Standard 57 87 88

byte substitution layer 90

diffusion layer 90

hardware implementation

key addition layer 90

selection process 88

software implementation

AES see Advanced Encryption Standard affine cipher 19

affine mapping Alice and Bob 4 anonymity

asymmetric cryptography see public key cryptography

brute force see brute force attack buffer overflow 11

authenticated channel see channel authenticated encryption

authentication tag

baby step giant step method Biham Eli 75 76

binary extended Euclidean algorithm birthday attack

diffusion 57 block ciphers

and hash functions Blowfish 81

brute force attack 7

for discrete logarithms BSI 22

CA see certification authority Caesar cipher see shift cipher cardinality see group Carmichael number

CBC see cipher block chaining mode CBC MAC

CC see Common Criteria CCM

certificate revocation list

certification authority CFB see cipher feedback mode chain of trust

challenge response protocol

Chinese Remainder Theorem chosen plaintext attack 27

cipher block chaining mode cipher feedback mode ciphertext 5

classified encryption 89 cleartext see plaintext CMAC

collision resistance

with block ciphers confusion 90

CRL see certificate revocation list CRT see Chinese Remainder Theorem cryptanalysis 3 9

implementation attacks 10

social engineering 10 cryptographic checksum see message

authentication code cryptography 2 3

CSPRNG see random number generator cryptographically secure

CTR see counter mode cyclic group see group

Data Encryption Standard 55

PC 1 permutation 67

PC 2 permutation 68

analytical attacks 75

COPACOBANA code breaking machine 74 cracker 73

Deep Crack code breaking machine 73 differential cryptanalysis 75 exhaustive key search 73

final permutation 61

hardware implementation 77

initial permutation 61

linear cryptanalysis 75

data origin authentication decryption exponent

DES see Data Encryption Standard DESX

deterministic encryption RSA

deterministic encryption stream ciphers 48 DHAES

DHKE see Diffie Hellman key exchange DHP see Diffie Hellman problem differential cryptanalysis 66

Diffie Hellman key exchange

Diffie Hellman problem

digital signature

Digital Signature Algorithm key generation

Digital Signature Standard Diophantine equation Dirichlet s drawer principle

discrete logarithm problem elliptic curves ECDLP

divide and conquer attack DLP see discrete logarithm problem Dobbertin Hans

for Diffie Hellman key exchange double encryption

DSA see Digital Signature Algorithm DSS see Digital Signature Standard

ECB see electronic code book mode

ECDH see elliptic curve Diffie Hellman key exchange

ECDHP see elliptic curve Diffie Hellman problem

EDE see encryption decryption encryption EEA see extended Euclidean algorithm electronic code book mode

Elgamal digital signature

acceleration through precomputation key generation

Elgamal encryption scheme elliptic curve

elliptic curve Diffie Hellman key exchange

elliptic curve Diffie Hellman problem elliptic curve Digital Signature Algorithm see

elliptic curves Koblitz curves

Encoding Method for Signature with Appendix see EMSAS

encryption exponent

encryption decryption encryption

equivalence class 15

Euclid s algorithm see Euclidean algorithm Euclidean algorithm

Euler s phi function Euler s theorem

exhaustive key search see brute force attack existential forgery

Elgamal digital signature RSA digital signature

square and multiply algorithm sliding window algorithm

extended Euclidean algorithm extension field

irreducible polynomial 97

polynomial arithmetic 95

fault injection attack Feistel network 58

Fermat s Last Theorem

Fermat s Little Theorem field

extension see extension field order 93

fingerprint of a message finite field 90

flip flop see linear feedback shift register

Galois Counter Mode Galois fields see finite fields Gardner Martin

gcd see greatest common divisor

GCHQ see Government Communications Headquarters

GCM see Galois Counter Mode generalized discrete logarithm problem generator see group

Government Communications Headquarters

greatest common divisor 17

primitive element group order see group Grover s algorithm

compression function

cryptographic hash functions

from block ciphers hash value

Hasse s bound see Hasse s theorem Hasse s theorem

HMAC see message authentication code hybrid protocols

hyperelliptic curve cryptosystems hyperelliptic curves

identity based cryptosystems IEEE i 87

implementation attacks see cryptanalysis implementation attacks

index calculus algorithm

initialization vector 48 in CBC mode

integer factorization problem

integrity inverse

IV see initialization vector

KDC see key distribution center kdf see key derivation function KEK see key encryption key Kerberos

Kerckhoffs principle 11

key derivation function key distribution center key distribution problem key encryption

key encryption key key establishment

key predistribution

key whitening 78

keyed hash function see message authentica tion code

Lagrange s theorem

lattice based public key schemes letter frequency analysis see substitution

LFSR see linear feedback shift register lightweight ciphers 78

linear congruential generator 35 linear feedback shift register 41

feedback coefficients 43

known plaintext attack 45 maximum length 44

linear recurrence see LFSR Lucifer 56

MAC see message authentication code

MAC secret prefix MAC secret suffix malleable

Elgamal encryption

man in the middle attack

McEliece cryptosystems

MDC 2 hash function meet in the middle attack

Merkle Damga rd construction

message authentication message authentication code

message digest see hash function message expansion factor Miller Rabin see primality test

MIM see man in the middle attack MISTY1 82

MMH see multilinear modular hashing modulo operation 14

MQ public key schemes multilinear modular hashing

multiparty computation 21

multiplication table

multivariate quadratic public key schemes

National Institute of Standards and Technology 88

National Security Agency 56 89

NIST see National Institute of Standards and Technology

NSA see National Security Agency

OFB see Output Feedback Mode OMAC

one way function

hash functions and one wayness order see group

OTP see One Time Pad

out of band transmission Output Feedback Mode

RSA digital signature

padding for block cipher encryption parallelization of encryption

perfect forward secrecy PFS see perfect forward secrecy physical security

pigeonhole principle

Pohlig Hellman algorithm for discrete logarithms

Pollard s rho method

post quantum cryptography

preimage resistance

Miller Rabin

probabilistic test prime

prime number theorem

generalized Mersenne

PRNG see random number generator pseudorandom

probabilistic encryption Probabilistic Signature Scheme PSS see

RSA digital signature product ciphers 57 provable security

public key cryptography public key infrastructure see PKI

quantum computer 88

rainbow tables random number generator

cryptographically secure 36 for prime generation pseudorandom 35

relative security 38

Rijndael see Advanced Encryption Standard and hash functions

Rivest Ronald Rivest Shamir Adleman see RSA round key 67

Chinese Remainder Theorem decryption

short public exponent side channel attacks

RSA digital signature attacks

Probabilistic Signature Scheme PSS

second preimage resistance

secret key see cryptography symmetric secure channel

Secure Hash Algorithm see SHA security

short term 12 security by obscurity 11 security level

security objectives

Shanks Algorithm

Shor s algorithm side channel attacks

simple power analysis single point of failure

single key see cryptography symmetric Skype 87

small subgroup attack

social engineering see cryptanalysis social engineering

SPA see simple power analysis square and multiply

square and multiply algorithm see exponenti ation

station to station protocol

stream cipher 30 31

STS see station to station protocol subgroup

substitution attack

substitution cipher 6

brute force attack 7

letter frequency analysis 8

symmetric key see cryptography symmetric

time memory tradeoff discrete logarithms

time memory tradeoff attacks timelinesss

triple DES 55 78 effective key length

TRNG see random number generator true trusted authority

unconditional security 36

warm up phase 48 web of trust Wi Fi 87

Williamson Graham

zero knowledge proofs 21