ECRYPT II

ICT-2007-216676

ECRYPT II

European Network of Excellence in Cryptology II

Network of Excellence

Information and Communication Technologies

# D.SPA.7
# ECRYPT2 Yearly Report on Algorithms and Keysizes (2008-2009)

Due date of deliverable: 31. July 2009
Actual submission date: 27. July 2009

Start date of project: 1 August 2008 Duration: 4 years

Lead contractor: Katholieke Universiteit Leuven (KUL)

Revision 1.0

| **Project co-funded by the European Commission within the 7th Framework Programme** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission services) | |

# ECRYPT2 Yearly Report on Algorithms and Keysizes (2008-2009)

**Editor**
Nigel Smart (BRIS)

**Past and present contributors**
Steve Babbage (VOD), Dario Catalano (Catania), Carlos Cid (RHUL), Benne de Weger (TUE), Orr Dunkelman (KUL), Christian Gehrmann (ERICS), Louis Granboulan (ENS), Tanja Lange (RUB), Arjen Lenstra (EPFL), Chris Mitchell (RHUL), Mats Näslund (ERICS), Phong Nguyen (ENS), Christof Paar (RUB), Kenny Paterson (RHUL), Jan Pelzl (RUB), Thomas Pornin (Cryptolog), Bart Preneel (KUL), Christian Rechberger (IAIK), Vincent Rijmen (IAIK), Matt Robshaw (FT), Andy Rupp (RUB), Martin Schläffer (IAIK), Serge Vaudenay (EPFL), Michael Ward (MasterCard)

27. July 2009
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

# Contents

**Abstract**

This report contains the official delivery D.SPA.7 of the ECRYPT2 Network of Excellence (NoE), funded within the Information Societies Technology (IST) Programme of the European Commission's Seventh Framework Programme (FP7).

The report provides a list of recommended cryptographic algorithms (e.g. block ciphers, hash functions, signature schemes, etc) and recommended keysizes and other parameter settings (where applicable) to reach specified security objectives. Due to possible advances in cryptanalysis, the report is revised on a yearly basis for the duration of the project. The report reflects state-of-the-art in public knowledge at the time of writing. It builds upon an earlier report produced by the ECRYPT NoE from the Sixth Framework Programme (FP6).

The fact that a specific algorithm or variant thereof is not included in this report *should not* be taken as indication that particular algorithm is insecure. Reasons for exclusion could just as well be limited practical use (e.g. lack of standardization and/or implementation), maturity, etc.

# Chapter 1

# Introduction

To protect (information) assets in an IT system, cryptographic protocols, algorithms, and keys are used to reach certain security objectives deemed necessary for the protection of said assets. What characterises today's approach is to rely on standardised, open security frameworks which are configured by "appropriate" algorithms, keys (and other parameters) to reach the security level as defined by the security objectives. Well-known examples of such frameworks are IKE, IPsec, TLS, S/MIME, etc. This is an important step forward from earlier approaches based on proprietary algorithms and protocols, kept (to the largest extent possible) unknown to the general public.

While it is recognised that the openness principle is the right way to go, it still does not make the problem of implementing security a trivial task since skill is still needed to determine which algorithms and keys are appropriate for the security objectives at hand. First of all, a better-safe-than-sorry approach, e.g. encrypting each message four times with different algorithms and huge key sizes, may not be the way to go, since it is likely to lead to bad performance, complex management, and in some cases bad "dependencies" between algorithms that could actually reduce security. As we will argue in more detail later, security is a process, rather than a state. In particular, the openness is sometimes a two-edged sword, leading to various attacks becoming known, and the average user may have difficulty in keeping up with cryptanalytic advances, *maintaining* a secure configuration of his/her system over time. Moreover, it is not always easy to understand the effects of *combinations* of different configuration options, e.g. what is the overall security level when protecting a $k$-bit AES key by an $n$-bit RSA key, using RSA padding option $x$?

The purpose of this report is to provide comprehensive, yet easy to use recommendations for the use of cryptographic algorithms, keys (key sizes), and other parameters in protocols such as those mentioned above. Specifically, the report contains the official delivery D.SPA.7 of the ECRYPT2 Network of Excellence, funded within the Information Societies Technology (IST) Programme of the European Commission's Seventh Framework Programme (FP7). While trying to keep the recommendations simple, we also at the same time provide extensive references for readers who are more technically oriented and would like to have more background information.

Since the field of cryptology is constantly advancing, the recommendations in this report are updated on an annual basis.

The report is organised as follows. We conclude this chapter by comparing this report's role relative to other similar reports. In Chapter 2 we give rationale for algorithm selection

criteria. Next, in Chapter 3 we introduce some notation and abbreviations used throughout the report. Then, in Chapter 4, we discuss security objectives on a high level, approaches to implement security relative to different type of attackers, and the importance of non-cryptographic issues, that are outside the scope of this report. The rest of the report is divided into three parts. Part I provides key size (and other parameter) recommendations. Part II provides recommendations on the use of symmetric algorithms, and Part III finally treats asymmetric algorithms.

## 1.1 Related Work

This is not in any way the first report containing recommendations for algorithms and key sizes. We will later survey previous work in more detail, but for the moment, we give a quick overview.

In [25], recommended key sizes for *symmetric* algorithms are given in relation to state-of-the-art in 1996. More recently, [117, 161] gives recommended key lengths for different attack scenarios and provides symmetric/asymmetric size-equivalence relations. A somewhat simpler version of the analysis in [117], geared toward the specific application of finding asymmetric sizes equivalent to (in practice fixed) symmetric sizes, can be found in [114, 115]. These reports are quite generic and do not provide algorithm recommendations. In [208] a survey and comparison of various key-size recommendations can be found.

The US NIST recommends algorithms and key-sizes for US federal use through the publication of FIPS (Federal Information Processing Standard) documents—an algorithm/key-size appearing in a FIPS is thus considered "approved". NIST also develops guidelines/recommendations in the form of Special Publications (SP).

The NESSIE consortium, [155], presents a portfolio of recommended algorithms, and in some cases, also key-size recommendations. Some of the algorithms in [155] have since been included in standards, some of which are covered here.

RSA Laboratories in [171] provide a cost-based analysis of equivalences between symmetric and asymmetric key sizes.

Finally, in [57], both algorithm and key-size recommendations can be found, but only for signature schemes.

In comparison to the above works, the current report aims at a wider scope, covering most types of algorithm primitives, in combination with recommendations for appropriate key sizes and the setting of other parameters. A second goal is to provide some easy to understand discussion on (security) properties of various primitives without using too technical language, yet striving to be correct. Indeed, experts in the area may find the text too informal in places (or even naive), so a third goal has been to provide numerous references to more technical treatments. As mentioned, the (at least) annual updates of this report aim to provide up-to-date information.

It is also important to distinguish between the questions *what is the smallest secure key size?*, and, *what is the smallest key size that is not completely insecure?* That is, rather than trying to find a magic "threshold" that separates security from insecurity, we have tried to be realistic and discuss more in terms of security levels, and elaborate considerations that could make a lower security level, forced onto us by environmental considerations in the form of bandwidth or processing power, acceptable. Also, the duration of the required protection is important, as is the value of the protected information asset.

# Chapter 2

# Algorithm Selection Criteria

The report provides a list of cryptographic algorithms in the following categories.

- Symmetric primitives:
  - block ciphers (and modes of operation)
  - stream ciphers
  - hash functions
  - message authentication codes.
- Asymmetric primitives:
  - public key encryption
  - signature schemes
  - public key authentication/identification schemes
  - key encapsulation mechanisms
  - key agreement and key distribution.

Why are some algorithms included, yet other (sometimes well-known) algorithms excluded? The basis for selection has been security and wide-spread use. As a principle, only standardised, mature, wide-spread and *secure* algorithms are included. However, in some cases, commonly used, but security-wise non-optimal algorithms are also covered, pointing out caveats in using them. There are also a few cases where draft standards, anticipated to have substantial near-future impact have been included.

Therefore, the fact that a specific algorithm or variant thereof is *not* included in this report cannot be taken as indication that that particular algorithm is insecure. Reasons for exclusion may as mentioned also be limited practical use (e.g. lack of standardization and/or deployment), etc. Conversely, inclusion does not guarantee that a particular algorithm is secure, only that it is secure as known in current state of the art. Current methods may fail, sometimes spectacularly, see e.g. [52].

Currently, the report does not cover pseudo-random functions, pseudo-random generators, entity authentication using symmetric techniques or more compound cryptographic protocols. In some cases "variants" of a basic algorithm, e.g. different RSA padding mechanisms, are covered due to their wide-spread use and/or expected advantages.

# Chapter 3

# Preliminaries

## 3.1 Notation

Throughout the report, the following notation is used.

| | |
|---|---|
| $\log_b x$ | logarithm of $x$ to base $b$ |
| $\log x$ | $\log_2 x$ |
| $\ln x$ | $\log_e x$, $e = 2.7182\ldots$ |
| $\lvert x \rvert$ | the size (in bits) of $x$, i.e. $\lceil \log x \rceil$ |
| $\mathbb{Z}_m$ | the ring of integers modulo $m$ |
| $\mathbb{Z}_m^*$ | the multiplicative group $\subset \mathbb{Z}_m$ |
| $\mathbb{F}_s$ | the finite field of $s = p^n$ elements |

Please refer to Appendix A for a glossary of abbreviations and acronyms.

We shall for each algorithm make references to various international standards making use of said algorithm. Though they are not standards in the formal meaning, the probably largest "consumer" of cryptographic standards are the IETF RFC specification suites. Since we shall often refer to these, we here wish to clarify use of some acronyms once and for all.

**IPsec:** IP security protocol, RFC 2401, 2402, 2406.

**IKE:** Internet Key Exchange, RFC 2409.

**TLS:** Transport Layer Security (TLS), RFC 2246.

**S/MIME:** Secure MIME, RFC 3396, 3850, 3851.

**OpenPGP:** RFC 2440, 3156.

These are available from `http://www.ietf.org/rfc.html`.

### 3.1.1 Primitive Information

For each primitive type (block cipher, hash function, signature scheme, etc) we only give informal, intuitive definitions of what it means for the respective primitive to be "secure". A number of different security notions have been proposed and used in the literature, and it is beyond the scope of this report to investigate them more deeply, compare them, etc. When important for practical use, we shall highlight in slightly more detail what security

notion we refer to. For the interested reader, we generally refer to the excellent survey given in each chapter of the NESSIE security report, [156]. Although our definitions are only informal, we shall not make use of any non-standard terminology, so [156] or any text book on cryptography, e.g. [137], should be able to provide more background, if needed.

### 3.1.2 Algorithm Information

Each of the included algorithms in Chapter 8 through Chapter 17, are represented in the form of an *algorithm record*, having the following shape and meaning:

**Definition:** Reference to stable algorithm specification.

**Parameters:** Parameters characteristic of the algorithm such as supported key and block sizes, as well as any data required to unambiguously specify algorithm operation, e.g. group used to carry out arithmetic etc.

**Security:** Statement about the security according to state-of-the-art. In particular, for symmetric algorithms, the effective key size, or, "as claimed", meaning that no non-trivial attack is known. For asymmetric algorithms, where applicable, reference to proof of security and possible assumptions needed for the proof to hold.

**Deployment:** Reference to standards and/or products that use the algorithm.

**Implementation:** Pointers to "reference" implementation or test-vectors, if such are known to exist.

**Public analysis:** Reference to public analysis carried out, e.g. research papers and/or efforts such as NESSIE, Cryptrec, etc, where applicable.

**Known weakness:** Reference and short explanation of known weaknesses, should such be known.

**Comments:** Any additional information, e.g. caveats, pros/cons in using the algorithm, etc.

Note that the algorithm records are not necessarily exhaustive.

# Chapter 4

# Security Objectives and Attackers

We introduce security in IT systems to meet certain desired *security objectives*. Examples of well-known security objectives closely coupled to cryptography are *confidentiality, integrity*, and *non-repudiation*. The *security level* determines quantitatively to what extent these objectives need to be met, e.g. "how strong" confidentiality do we need, and is based on a threat and risk assessment of the IT system, which somewhat simplified asks questions of the following form:

1. How long into the future must security level persist? What is the "lifetime" and/or "value" (direct/indirect) of the protected asset?
2. Success of a brute-force key search is inevitable, but are there even better attacks?
3. What is the *attack model*? (Who is the attacker? What resources does he/she have?)
4. How will these resources develop during the lifetime of the protected data?
5. What cryptanalytic progress will occur during the lifetime of the protected data?

The approach taken in this report (and all other existing similar reports) is to take present state-of-the-art for the second point, make assumptions about the three last issues, and from that extrapolate key-sizes to match different security levels as defined by the first point. In the following we discuss in some more detail what this means.

Before doing so, a few words about defining the security level, i.e. (1) above. This is a business issue, figuring out what the overall financial impact of a security breach could be. Here, the more indirect value of "reputation" is typically a bigger consideration than direct monetary impact. This is then combined with the cost of various types (levels) of protections, and finally one decides what security level is desirable and till when.

## 4.1 The Security Process

It is important to realize that security is usually more of a process than a state. Security means to use and manage mechanisms to

**Protect:** make attacks expensive or unsuccessful.

**Detect:** make (attempted) attacks likely to be detected.

**Respond:** provide means to respond to successful attacks.

**Recover:** restore the system to a secure state after an attack.

Sometimes one also includes deterrence, which in a cryptographic sense is quite similar to protection (a large key size may deter attackers by the implied effort needed to break the system) but it is also covers non-technical issues such as legal frameworks to prosecute attackers. Such non-technical issues will not be discussed here.

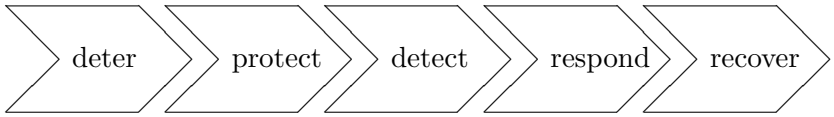


Figure 4.1: The security process, [29].

One can identify two main approaches in implementing this security process. The *fail-safe* approach puts a lot of resources on deter/protect to make security failures unlikely, whereas what we (with slight abuse of English language) shall call the *safe-fail* approach spends effort more on detect/protect/recover, so that failures cannot be excluded, but can on the other hand be recovered from. As we will see, different security objectives are sometimes more suited for one of these two approaches.

Part of the goal of this report is to provide some guidance on how to approach important parts of the security process in order to maintain a secure system. Still, many important aspects of the process are also out of scope, e.g. management processes surrounding cryptographic keys, such as revoking keys and replacing keys before they are "worn out" etc.

## 4.2 Security Levels

At the end of the day, what one usually cares about is how long it will take an attacker to "break" the security and what resources he needs in order to have a reasonable chance of succeeding. This cost (in time and/or money) of breaking the system must be higher than the value (in a similar measure) of the protected asset. For instance, if a 1Million dollar machine is needed to obtain secrets of value 1 dollar, one could hope the attacker is deterred by this (unless he/she simultaneously can get one million such secrets of course), and similarly, if information is only "sensitive" for a few hours, a break of the system requiring a week's attack effort *might* be acceptable.

It is however important to note that the needed security level could depend on the security objective. Referring to the security process above, if confidentiality is compromised, there is usually not much one can do to recover from the compromise. However, if we fear loss of non-repudiation to be close at hand, various combinations of recovery mechanisms may be available, e.g. re-signing with new algorithms and/or longer keys. Therefore, confidentiality is usually by nature a security objective that needs to be handled with the above defined fail-safe approach, whereas e.g. non-repudiation can sometimes also be handled by the safe-fail approach.

## 4.3 Management Aspects

When the security requirements are very high, it is usually the case that non-cryptographic issues become more important. For instance, to maintain an overall system security level of more than, say 128 bit symmetric keys, management and social aspects tend to weigh in more than mere key size. We do not claim that it is impossible or irrelevant to imagine very high security levels, but we want to stress that such non-cryptographic issues are then likely to enter the scene as the weakest link of the chain. This report therefore simply assumes that management issues can be handled to match the desired security. It is out of the scope of this report to convert such aspects into key sizes, attempting to answer a question like: "We use $k$-bit keys, managed by a process like this, what is the overall effective key size?"

## 4.4 Attack Resources and a Note on Moore's Law

What attackers are realistic to take into account? In 1996, [25] used the following classification.

**"Hacker":** using a $0 budget and standard PC(s), or possibly a few $100 spent on FPGA hardware.

**Small organization:** with a $10k budget and FPGA(s).

**Medium organization:** $300k budget and FPGA and/or ASIC.

**Large organization:** a $10M budget and FPGA/ASIC.

**Intelligence agency:** $300M budget and ASIC.

Such a classification is probably largely still valid today.

When security is to be maintained for longer periods than a few months, we must also take into consideration that the attacker may upgrade his/her resources according to developments in state-of-the-art. The sometimes debated, but commonly accepted way to handle this point is to assume Moore's law. This law, today usually (mis)quoted as

$$\text{bits per square inch in year } t \sim 2^{(t-1962)/1.5},$$

has been a reasonably good approximation for the developments in "computing power per dollar" over the last few decades. For example, [25], mentions implementation of a DES cracker on a certain family of FPGAs. The gate density of this family of FPGAs have increased from tens of thousands of gates (1995) to hundreds of thousands (2002).

Moore's formula is considered very inaccurate at describing CPU clock-speed, but for the probably more relevant measure of PC performance in terms of MIPS capacity, it seems to apply albeit with a slightly larger constant than "1.5", see [192] for a discussion.

Industry experts seem to agree that it is likely that Moore's law will continue to apply for at least a decade or more. Therefore we have chosen to adopt this assumption here (as is also done in e.g. [117] and previous key-size studies). However, completely new computational models and forms of hardware may also need to be considered, more on this later.

## 4.5 Implementation Notes

Cryptographic algorithms can be more or less sensitive to attacks that primarily exploit "physical" rather than cryptographic weaknesses. That is, attacks could be relying on implementation mistakes or exploitable environmental properties. For instance, some attacks are known based on e.g. a biased key-space, timing analysis, fault analysis, power consumption analysis, etc. Misuse and management errors can of course also lead to compromise. The recommendations in this report are given under the assumption that the algorithm is properly implemented, used, and managed, and furthermore run in an environment where it is not subject to the above mentioned side-channel attacks. Occasionally, however, we may point out specific caveats of this nature.

# Part I

# General Key Size Recommendations

# Chapter 5

# Determining Symmetric Key Size

For symmetric schemes, key size requirement is in principle quite straightforward. If the cryptosystem as such can be assumed to be secure for the lifetime of protected data, the only attack means is usually a brute force key search/guessing attack[1], whose time/success ratio only depends on the maximum amount of computing power (number of computers, special purpose hardware, etc), $P$, the foreseen attacker(s) have at their disposal. Thus, we select an $n$-bit key so that $\sim 2^n/P$ is somewhat larger than the life-time of the protected data. If the secret must prevail for more than, say a year, we should as mentioned also take Moore's law into account. Let us survey some earlier recommendations and some (publicly known) attack efforts, to benchmark how well extrapolation of old recommendations seem to hold.

Our basis will be the report [25], which in 1996 proposed 75-bit symmetric keys to protect "against the most serious threats", and 90-bits for a 20-year protection. The proposal is based on Table 5.1, which seems to be rationalized by defining minimum security as maintaining confidentiality a few days, up to a year. (The last column shows the assumed key-recovery time for the different attackers in the model used.) There are some indications that this is

Table 5.1: Minimum symmetric key-size in bits for various attackers (1996).

| Attacker | Budget | Hardware | Minimum keysize | Recovery time |
|---|---|---|---|---|
| "Hacker" | 0 | PC(s) | 45 | 222 days |
| | $400 | FPGA | 50 | 213 days |
| Small organization | $10k | FPGA | 55 | 278 days |
| Medium org. | $300k | FPGA/ASIC | 60 | 256 days |
| Large org. | $10M | FPGA/ASIC | 70 | 68 days |
| Intelligence agency | $300M | ASIC | 75 | 73 days |

still a reasonably accurate approach. We have some recent data-points of attack efforts, as well as some proposed hardware (FPGA and ASIC) designs to compare to.

[1] See considerations of trade-off attacks, Section 5.1.1.

## 5.1 PC/Software Based Attacks

In late 1999, a small team which for these purposes could be considered (skilled) "hackers" found a 48-bit DES key in about 3 weeks using a relatively small number of general purpose computers, distributing key-search as part of a screen-saver application, [5]. Applying Moore's law to Table 5.1, it predicts that 48-bit keys in 1999 would resist hackers for about a year. Thus, the attack was about 17 times faster than expected. On the other hand it is doubtful if [25] included such an attack in the "hacker" category, and we return to this later.

In 2002, [51] completed the RC5 64-bit challenge after nearly 5 years and some 300,000 individual general-purpose computers participating. (The same source completed the 56-bit challenge in 250 days in 1997 using 3,000 machines.) A question is how to interpret such results, since they are done by "hackers" with relatively small resources, utilizing the connectivity offered by today's (and tomorrow's) Internet. This is an important issue, since if one indeed considers this a hacker attack, the hacker keysizes need a quite big increase relative to predictions by the table (at least 5-10 bits). One could on one hand argue that such resources will only be summoned on "challenge" types of attacks. On the other hand, various forms of Malware (e.g. worms[2]) are more and more frequent, and the average user could, unbeknownst to him/her, be part of a real attack. Indeed, [184] reports that this is one of the most popular applications of worms.

In [53] it is estimated that the total computational power of the Internet is about $2^{85}$ operations per year. While it would unrealistic that the whole Internet spends one year to attack a single 85-bit key, one can consider more reasonable examples.

Suppose, for instance, that a "worm" with key-search motive is spread to a fraction $h$ of all hosts. Suppose also that the worm, to avoid detection, runs in "stealth" mode, consuming a $c$-fraction of the CPU power of each host. (We assume for simplicity that all hosts have equally powerful CPUs.) Finally, assume it takes a $t$ fraction of a year before the worm is disabled by anti-virus means. Even for quite realistic values of $h, c, t$ (say around 0.1–1% of all hosts, below 1% CPU power and roughly a day before anti-virus has effect), we see that keys somewhere in the range 56–64 bits could be retrieved.

These case-studies leads us to believe that we need to introduce a new type of hacker using distributed attacks, perhaps stealing CPU time from victims at least on a medium scale, and we propose to add an additional 8 bits for this category. We shall not explicitly consider the very large scale challenge type efforts.

### 5.1.1 Time-Memory-Data Trade-offs

We have so far assumed brute force key-search to be the only possible attack. This is not necessarily true if, for instance, the following conditions hold

- The adversary can use an "off-line" pre-computation step, generating large amounts of data (e.g. encrypting messages under random keys).
- The adversary has large storage capacity.
- The adversary will be able to observe a large number of encrypted messages under different keys, and it suffices for him to break one of these keys.

[2] A piece of computer program that automatically spreads to hosts on a network, infecting them with "malware".

Attacks in this model are based on the adversary pre-generating data, storing it in a database and finding "lucky" collisions between observed data and data stored in said database. "Luck" is here typically dependent on the size of the key, the amount of pre-computed data, and the number of observed messages, and generalizations of the well-known birthday paradox. Table 5.2 from [53] summarizes some generic, algorithm independent attack complexities that can be achieved by such *time-memory-data* (TMD) trade-offs.

Table 5.2: Generic TMD attack trade-off points.

| Key-size | No. of keys (Data) | Time | Memory | Pre-processing |
|---|---|---|---|---|
| $n$ | $2^{k/4}$ | $2^{k/2}$ | $2^{k/2}$ | $2^{3k/4}$ |
| $n$ | $2^{k/3}$ | $2^{2k/3}$ | $2^{k/3}$ | $2^{2k/3}$ |
| $n$ | $2^{k/2}$ | $2^{k/2}$ | $2^{k/2}$ | $2^{k/2}$ |

For example, if the attacker has $2^{43}$ memory and $2^{85}$ pre-processing computations, he can, using $2^{84}$ time, obtain one 128-bit key among a set of $2^{43}$, and so on.

## 5.2 Attacks Using ASIC Designs

EFF's DES-craker (a.k.a. "Deep Crack"), [63], designed and built at a cost of about US$200k, was reported to recover 56-bit DES keys in just over 22 hours in 1998. Table 5.1 predicts that in 1998, keys of roughly 61 bits would resist such attackers for 2 days. Extrapolating, Deep Crack would have retrieved such keys in about 1 month. Table 5.1 is thus slightly conservative, but close.

In [31] (1999), a $280M machine is sketched that would occupy an area one fifth of the Pentagon and would retrieve 64-bit RC6 keys in a few minutes. It is noted that this figure matches quite well an extrapolation of [25] numbers.

More recently, in [49] (2006) an ASIC design targeted at hardware-focused stream ciphers is suggested. The idea is that by definition, such stream ciphers are easy to accelerate with hardware (and hardware replication), speeding up key-search. Also, some intrinsic stream cipher properties can be exploited, e.g. one would expect that half of the keys can be discarded already after only one bit of output has been generated. This paper suggests e.g. that 80-bit (stream cipher) keys could in 2010 be found in one month using $33M of hardware. (Alternatively, with a $8M budget *today* the key could be retrieved in a year.) This agrees reasonably well with the 1996 predictions of [25], which suggests that in 2010, 80 bit keys could be found at third this cost, but taking twice the time, i.e. a similar cost/time trade-off.

## 5.3 Attacks Using FPGA Designs

In the following, we will estimate the costs for an exhaustive key search attack on the symmetric ciphers DES and AES based on the use of FPGA devices. As basis for our estimates, we use area-time optimized FPGA implementations of DES and AES presented in the next section. As main result of our assessment we conclude that a DES key search using FPGAs is very feasible at low costs and moderate engineering expertise. In the case of AES, however, a brute force attack seems completely out of reach for several decades to come.

The reason why we focus in more detail on FPGAs as opposed to ASICs (Application Specific Integrated Circuits) is that that FPGAs require (1) considerably less engineering expertise and (2) the initial costs for an FPGA development environment are very moderate (perhaps a few 1000s of dollars vs. 100,000s of dollars in the ASIC case, see, e.g. [56]). We would like to stress, though, that ASICs are more cost efficient than FPGAs in high volume applications. Thus, in the case of key search machines for AES, one could possibly achieve lower costs with dedicated ASICs. However, the expected cost/performance benefit should be not more than 1–2 orders of magnitude, so that even an ASIC-based key search attack seems completely out of reach.

### 5.3.1 Existing Area-Time Efficient FPGA Implementations

A comparison of different AES implementations on FPGAs can be found in [76]. Table 5.3 summarizes relevant information and the corresponding references.

Table 5.3: Comparison of some published FPGA AES implementations (128 bit key)

| Design | Device | Slices | Block RAM | Through-put (Gbit/s) | Mbit/s / Slice |
|---|---|---|---|---|---|
| Gaj et al. [60] | XCV1000-6 | 12600 | 80 | 12.1 | 0.96 |
| McLoone et al. [132] | XCV812E-8 | 2222 | 100 | 6.95 | 3.1 |
| Standaert et al. [188] | XCV3200E-8 | 2784 | 100 | 11.77 | 4.2 |
| Saggese et al. [173] | XVE2000-7 | 5810 | 100 | 20.3 | 3.4 |
| Hodjat et al. [76] | XC2VP20-7 | 5177 | 84 | 21.54 | 4.0 |

Apparently, [188] and [76] yield the best performance of AES on an FPGA regarding the area-time trade off. A throughput of 11.77 Gbit/s and 21.54 Gbit/s could be reached, respectively. Another contribution shows efficient FPGA accelerators both for DES and AES ([105]). For a throughput of 744 MByte/s, DES requires 3,250 CLBs (Configurable Logic Blocks) on a single FPGA. For a throughput of 2,500 MByte/s, AES requires 5,350 CLBs and 80 Block RAMs.

Besides results from the research community, there are also commercial cores for symmetric ciphers available, e.g., from Helion. The “Helion AES Core” runs on a low-cost FPGA (Xilinx Spartan-3) and is available in four different gradings: a “Tiny Core” with a throughput of 18 Mbit/s, a “Standard Core” with 229 Mbit/s, a “Fast Core “ with approx. 1 Gbit/s, and a “Pipelined Core” with more than 10 Gbit/s ([75]).

For DES, Helion offers a core running on a Xilinx FPGA (Virtex-II) with a data rate greater than 640Mbit/s for single DES and more than 230Mbit/s for triple DES ([75]).

### 5.3.2 Exhaustive Key Search Based on FPGA Hardware

For an exhaustive key search, all possible keys are tested with the FPGA hardware implementations of DES or AES. Assuming the possession of a pair clear-cipher text, we only need to encrypt a single block with each key candidate. All computations are based on the results of [188, 105], where non-pipelined implementations are used (pipelining cannot be used since we only encrypt/ decrypt a single block with one specific key).

For AES, we can test approximately $9.2 \cdot 10^7$ keys per second. With the DES hardware, $1.2 \cdot 10^7$ keys per second can be checked. Hence, with a single encryption or decryption unit, an exhaustive key-search would take on average $3.1 \cdot 10^9$s ($\approx$ 98 years) and $1.9 \cdot 10^{30}$s ($\approx 5.9 \cdot 10^{22}$ years) for DES and AES (128 bit key), respectively. These numbers are already a first indication of the feasibility of an exhaustive key search for DES vs. AES.

### 5.3.3 Cost Estimates

In order to achieve low over-all costs, we assume high volume prices for FPGAs. For instance, a typical low-cost FPGA such as the Xilinx Spartan-3 FPGA (90nm process) currently costs about $6.50 ([209]) and contains approximately 8,000 configurable logic blocks and, hence can fit two AES or two DES units including overhead. Since the FPGAs listed in Section 5.3.1 differ from the Xilinx Spartan-3 platform, the following cost calculations are not exact but should give a valid estimate.

Assuming an exhaustive key search within one month, many FPGAs have to be assembled in a cluster, each performing key searches in a part of the key space. We assume an overhead of 100% for printed circuit boards, power supplies etc. Costs for development, electrical energy and possibly cooling are not included in this estimate. Under these conditions, DES can be cracked in one month with 75 FPGAs (containing two DES engines each), at a cost of $13 (FPGA plus 100% overhead), yielding a total cost of approximately $1,000.

Note that these numbers are based on an average key search, i.e., half the key space is searched. The cost-performance ratio stays constant in this estimation. That means, for instance, that a ten times faster key search (i.e., 3 days) would require a key search machine which is ten times more expensive (i.e., $10,000). Again, we would like to stress that these are estimates rather than exact numbers.

For AES (128 bit), a machine performing an average key search in one month would cost $\$4.6 \cdot 10^{24}$. Even if we take Moore's Law into account (i.e., decrease of IC costs by a factor of two every 1.5 years at constant performance), such a machine would still cost $\$4.2 \cdot 10^{14}$ in the year 2055. It is, however, somewhat doubtful if Moore's Law will be valid for such a time span, as semiconductor-based computers will run in various physical limitations before that. However, this is also a cost related issue: even if technology gets stuck, manufacturing capacities may still increase and costs decrease, thus keeping Moore on track, possibly.

### 5.3.4 Research in the Field of FPGA-Related Attacks against DES

A paper by Hamer et al. ([69]) describes a DES-cracking hardware on a field-programmable system called the "Transmogrifier 2a". A fully implemented system would be able to search the entire key space in 1040 days at a rate of 800 million keys/second. A single unit consists of two Altera 10K100 FPGAs, each connected to up to 4MB memory. The whole system consists of 16 such units and was expected to cost approximately $60,000.

Besides a simple exhaustive key search machine, [111] presents a hardware FPGA implementation of linear cryptanalysis of DES. The authors implemented a modified attack from Matsui's original attack. The resulting attack is less efficient than Matsui's attack, but fits in the hardware and breaks a DES key in 12-15 hours on one single FPGA.

A quite accurate estimate of an attack on DES by an exhaustive key search is given in [187]. Assuming the use of low-cost devices such as the 3S1000 Spartan-3 from Xilinx ($

12 p.p.) and recent results from efficient DES implementations, a device for cracking DES within 3 days would cost approximately $ 12,000.

In [113] (2006) a €9,000 FPGA design, "COPACOBANA", is made that retrieves 56-bit DES keys in about nine days, average. This is slightly (but not by any order of magnitude) slower than extrapolated from the above DES machine. Subsequently, [50] proposes a similar design targeted at stream cipher key search.

## 5.4 Conclusions

Motivated by the analysis above, our proposal is to (according to Moore's Law) add about 8 bits to all the figures in Table 5.1, and to add the distributed attack type as above with some additional safety margin. ECRYPT's symmetric key-size table is found in Chapter 7.

Let us do a quick "sanity check" by comparing the above conclusion to the work in [117]. That study is based on a key size requirement defined in terms of Moore's Law and that 56-bit (DES) keys were "sufficiently infeasible" in terms of a 1982 $50M machine retrieving 56-bit keys in 2 days. Now, for 2008 [117] recommends symmetric keys of about 75 bits. A Moore-extrapolation (to add about 8 bits to the 1996 [25] values) here leads to a (2008) $300M machine finding 83-bit keys in 73 days. It is tempting to turn the ($50M, 75-bit, 2-day) machine suggested by [117] for 2008 into a $300M machine that recovers (almost) 78-bit keys in the same time, or, 83-bit keys in about 64 days. This thus compares well with our estimate.

For FPGAs, Table 5.1 predicts that in 2008, 58-bit keys will give about 200 days of protection against a $400 FPGA design. Above, we estimated that $12,000 would give us a device that retrieves such key-sizes in 3 days. Scaling, $400 spent on FPGA would give us the 58-bit keys if we are willing to wait 30 times longer, i.e. 90 days.

### 5.4.1 Side Channel Attacks

While previous considerations dealt with the algorithmic strength of the cryptographic primitives, a different research field analyzes the strength of the implementation itself. This approach is motivated by the fact that even strong primitives can be implemented in a way that is easily vulnerable to an attacker. Such *side channel attacks* actually find the key using information which leaks from the execution of the encryption, whether it is different timings for different keys/operations, different power consumption curves, or even different memory accesses.

Timing attacks [109] are attacks which are based on observing the time required to perform the secret operation. For example, consider an RSA decryption done using an unprotected implementation of the square-then-multiply algorithm. In such a case, the time required for a step of the decryption process when the bit of the secret exponent is 1 is about twice as much as the case when this bit is 0. By observing the time required for the decryption process, it is possible to obtain information about the hamming weight of the secret exponent. Also, by using specially crafted ciphertexts and statistical methods, it is possible to retrieve the entire key.

Power analysis [110] follows similar concepts of observing internal information on the execution of the algorithm, but uses the power consumed by the implementation. For example, the power consumption when flipping a memory bit from 0 to 1 (or from 1 to 0) is significantly higher from the power for maintaining a 0 bit (using standard logic gates). This attack is

especially useful against devices which use external source of power (e.g., smart cards and RFID tags), where the attacker can easily monitor the amount of power consumed by the device. A great deal of research has been put into offering protection against this type of attack, but usually new protection methods do not withstand the new generation of these attacks.

Cache attacks [163] are a special case of timing attacks. While regular timing attacks are usually very successful against asymmetric primitives, they usually fail against symmetric key algorithms, as their execution time is very close to a fixed time (and the execution time is not correlated enough with the actual values that are encrypted). The cache attacks overcome this issue by measuring the state of the cache (through timing) before and after encryption steps. By observing which entries have been accessed, a very fine tuned information on the encryption process can be detected. Recent works on this topic have suggested the use of the TLB (Translation lookaside buffer) of modern CPUs. It is worth mentioning that recent results on AES in the HyperThreading computational model [162] has led Intel to include AES commands in future CPUs.

Various countermeasures have been suggested to overcome these attacks. Some of these ideas are generic approaches (e.g., maintaining a shadow value which is encrypted in a manner which cancels the difference in power consumption) and some of them are algorithmic specific. While the exact analysis of the primitive against these attacks is usually hard to anticipate (as the attack is on a specific implementation, and do not interact directly with the algorithm), there are some basic observation which are true: The use of table lookups increase the susceptibility of the cipher to cache attacks (as the attacker can try and observe which of the table entries were accessed). The simpler the operation is, it seems easier to protect it (for example, ensuring that the power usage of an XOR operation is constant is much easier than ensuring that the power usage of a 32-bit multiplication is constant).

Thus, when coming to pick the right cryptographic solution it is important to define the mode of use, and the access that the attacker might have to the implementation. The stronger the attacks the attacker may mount as side channel attacks, the higher the cost of protecting the implementation becomes.

# Chapter 6

# Determining Equivalent Asymmetric Key Size

For asymmetric cryptography, things are more complicated since

- unnecessarily large keys affect performance
- increasingly effective attacks (all *much* better than brute force) have been discovered over the last 30 years
- we have recently seen suggestions for special purpose cryptanalytic hardware that are more than being "naive" search-machines.

Symmetric and asymmetric schemes are often used in conjunction, e.g. an asymmetric key is used to guard a symmetric key. Thus, we must be aware of the somewhat worn-out, yet universally valid "security is as good as the weakest link" principle, and we need a way to find an asymmetric key-size that matches a pre-specified symmetric key-size. At the same time, we want to avoid an overly large public key. A number of comparative studies have been made by skilled people, and rather than doing it again, we would like to see what conclusions can be drawn from these sources.

## 6.1 Inter-system Equivalences

In the following, *RSA* refers generically to a public key scheme, which is here assumed equivalent in hardness to the integer factoring problem. We similarly use *DLOG* and *EC* to refer generically to schemes based on discrete logarithms and (secure) elliptic curve groups, respectively, both over finite fields of prime order.

For EC, state-of-the-art suggests that EC in a subgroup of $m$-bit size is equivalent to $m/2$ bits symmetric key. In fact, considering that the elementary operations of these attacks are point additions on an elliptic curves, which should be compared to trying a symmetric key, it would be possible to reduce EC key size by roughly 8-12 bits for practical ranges of key-sizes. This is done in [117], but not in e.g. [149, 114]. We propose to follow the simple half-the-size principle as a general recommendation. This will also, as noted in [117], provide some protection against unforeseen developments in cryptanalysis of elliptic curves, without serious impact on performance. Curves over binary fields are often recommended (see e.g. [207]) to use slightly larger keys to mitigate certain hardware attacks. Also, the generic attacks are

in the binary case sometimes a bit more efficient. For instance, so-called Koblitz curves over binary fields of size $2^n$ can be attacked roughly a factor $\sqrt{2n}$ faster. Curves over binary fields generally also require other forms of special attention when selecting parameters.

According to state-of-the-art, the difficulty of solving DLOG in prime order fields of size $2^n$ is, up to constants, asymptotically equivalent to that of breaking $n$-bit RSA. In practice though, DLOG is noticeably more difficult. Moreover, DLOG is in most standardized algorithms performed in a smaller *subgroup*, and then the size of this subgroup that matters too, in which case the symmetric key equivalent is theoretically half of the bit-size of said subgroup, again according to the same generic attacks applicable also in the EC case. This would imply DLOG sub-groups of the same size as a corresponding EC group. Note though that performing exponentiations over a finite field is noticeably more expensive than on an elliptic curve of equivalent security. (The difference can be on the order 10-40 times, depending on security level.) Implementations which are concerned with performance could therefore, if required, reduce subgroup size by a few bits without lowering the security compared to the EC case (e.g. [117, 161] uses roughly 10-bits size reduction of the subgroups for practical key-sizes). Nevertheless, our general recommendation shall for simplicity be according to the same half-the-size principle, and to assign $n$-bit fields the same security as $n$-bit RSA, a slightly conservative recommendation for DLOG based schemes.

With this in mind, the main issue is to determine RSA vs. symmetric key equivalence. We next survey some existing publications in this area.

## 6.2 Survey of Existing Guidelines

Assuming that the 56-bit Data Encryption Standard (DES) was considered "secure" by any reasonable meaning until 1982, [117] suggest that 80-bit symmetric keys are secure until 2012 and are *computationally* equivalent to 1120-1464 bit RSA/DLOG keys (depending cost model; HW/general purpose computer), 140 bit DLOG subgroups and 149-165 bit EC groups (depending on likelihood of cryptanalytic progress). The analysis is quite methodical, and can be used to derive key-sizes relative to varying assumptions and scenarios.

Table 6.1: Lenstra-Verheul recommendations (assuming "average" cost model/cryptanalytic progress).

| Equivalent symmetric key size | 56 | 64 | 80 | 96 |
|---|---|---|---|---|
| Elliptic curve key size | 95 | 116 | 160 | 200 |
| Modulus length ($pq$)/dlog field | 380 | 580 | 1300 | 2500 |
| Dlog subgroup | 102 | 114 | 141 | 171 |

Taking *cost* (of memory) more into consideration, [183] argues that in the year 2000, 1024-bit RSA keys corresponded to 96-bit symmetric keys. As noted in [180], this leads to quite different extrapolations of the lifetime of RSA keys between [117] and [183], the second giving 1024-bit RSA keys some 30 years extra lifetime. On the other hand, worst-case scenarios in [117] suggests that 1024-bit RSA keys could already have been factored today, but there is no public record that any key even close in size has been factored.

In the US, NIST has issued an equivalence table, [149], stating that 80-bit symmetric keys correspond to 1024-bit RSA/DLOG keys and 160-bit DLOG subgroups and EC groups. For

the 128-bit level, 3072 and 256-bit keys, respectively, are recommended. The 80/1024/160 bit level is considered sufficient until 2010, beyond that 112/2048/224 bits are promoted. RSA

Table 6.2: NIST recommendations.

| Equivalent symmetric key size | 80 | 112 | 128 | 192 | 256 |
|---|---|---|---|---|---|
| Elliptic curve key size | 160 | 224 | 256 | 384 | 512 |
| Modulus length ($pq$)/dlog field | 1024 | 2048 | 3072 | 7680 | 15360 |
| Dlog subgroup | 160 | 224 | 256 | 384 | 512 |

Lab's and Certicom's recommendations are in line with this. For all practical purposes, this is also in reasonable agreement with [117] but does not differentiate the cost of operations in EC and DLOG subgroups, compared to symmetric key operations.

Recently the NESSIE consortium, in [156], for "medium term" (5-10 years) security recommends the use of 1536-bit keys for RSA and DLOG based public key schemes, and 160-bit for elliptic curve discrete logarithms, suggesting a 1536/80 equivalence, which is in line with [117], with the same cost-remark for EC as above. This recommendation is based on an as-

Table 6.3: NESSIE recommendations.

| Equivalent symmetric key size | 56 | 64 | 80 | 112 | 128 | 160 |
|---|---|---|---|---|---|---|
| Elliptic curve key size | 112 | 128 | 160 | 224 | 256 | 320 |
| Modulus length ($pq$) | 512 | 768 | 1536 | 4096 | 6000 | 10000 |
| Modulus length ($p^2q$) | 570 | 800 | 1536 | 4096 | 6000 | 10000 |

sumed equivalence between 512-bit RSA keys and 56-bit keys, and an extrapolation of that. However, it should be noted that during the course of producing this report, we discovered that the NESSIE recommendations were based on a slightly inaccurate extrapolation formula, leading to somewhat too large RSA keys.

RSA Labs, in [171], performs a cost-based analysis, and arrives at the following key-size equivalences (Table 6.4). The time to break is computed assuming a machine that can break

Table 6.4: RSA Labs analysis.

| Symmetric Key | EC | RSA | Time to Break | Machines | Memory |
|---|---|---|---|---|---|
| 56 | 112 | 430 | $< 5$ min | 105 | trivial |
| 80 | 160 | 760 | 600 months | 4300 | 4Gb |
| 96 | 192 | 1020 | $3 \cdot 10^6$ yrs | 114 | 170Gb |
| 128 | 256 | 1620 | $10^{16}$ yrs | 0.16 | 120Tb |

a 56-bit DES key in 100 seconds, then scaling accordingly. The Machines column shows how many NFS sieve machines can be purchased for $10 million under the assumption that their memories cost $0.50/Mbyte[1].

The IETF recommendation RFC3766 [161] (which is largely based on the cost-model variation of [117]) suggests that an 80-bit symmetric key is equivalent to 1228-bit RSA/DLOG

[1]The 0.16 entry at the 128-bit level signifies that only 0.16 of the needed 120Tb memory can be purchased for the available $10M budget.

keys, and 148-bit DLOG subgroups. Specifically, Table 6.5 is given. IETF also issued rec-

Table 6.5: IETF RFC3766 recommendations.

| Equivalent symmetric key size | 80 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| Modulus length $(pq)$/dlog field | 1228 | 1926 | 4575 | 8719 | 14596 |
| Dlog subgroup | 129 | 186 | 284 | 383 | 482 |

ommended lifetime for (short) RSA signatures, [206]. This is based on a 100-fold security

Table 6.6: Short RSA signature key lifetimes from [206].

| Keysize | Lifetime |
|---|---|
| 512 | 1 hour |
| 576 | 10 hours |
| 640 | 4 days |
| 704 | 30 days |
| 796 | 8 months |
| 1024 | 1 year |

margin, e.g. the 512-bit keys should require 5 days of attack effort.

Basically all of the above reports explicitly or implicitly discuss requirements for *confidentiality*. It is sometimes argued that requirements for *authenticity* can be lower since data can be re-authenticated at regular intervals with increasingly large keys and even new algorithms. For signatures, the ETSI report [57] "approves" key-sizes inline with those above: 1024-bit minimum factoring/finite field discrete log keys and 160-bit elliptic curve groups size, but the approval is to be re-evaluated after five years (i.e. the approval is valid for the period 2001-2005).

### 6.2.1 Approach chosen by ECRYPT

Let $n_{512}$ be the symmetric (e.g. DES) equivalent key-size to a 512-bit RSA key. Similar to NESSIE and motivated by the above discussion, ECRYPT's public key size recommendations for factoring/RSA based and DLOG based schemes are based on an estimate for $n_{512}$ and an extrapolation of the attack complexity of the general number field sieve algorithm (GNFS), the presently fastest method for computing integer factorization and discrete logarithms[2]. Our approach is, as mentioned, thus based on the assumption on equivalence between RSA and factoring[3].

Specifically, the run-time of GNFS to factor $N$ is estimated to (asymptotically) be

$$L(N) = Ae^{(C+o(1))(\ln N)^{1/3}(\ln \ln N)^{2/3}},$$

for a constant $A$, and $C = (64/9)^{1/3}$. We shall make the assumption that the $o(1)$-term can, for the key size ranges at hand, be treated as zero. From this, we can calculate $A$ based on $L(512) = n_{512}$.

[2]Current records are 200 decimal digits [12] for GNFS and 274 digits [11] for numbers of special form for which a special version, SNFS, applies.

[3]Some recent results [98] study situations under which RSA may, in fact, be easier than factoring.

This leaves us with the problem of determining the quantity $n_{512}$. Experience from available datapoints suggests that the "resistance" of RSA-512 is some $4-6$ bits less than that of DES-56. We here choose the more conservative end of this interval, i.e. $n_{512} = 50$.

We now get the following expression for the effective key-size of $n$-bit RSA modulus $N$:

$$s(n) = \left(\frac{64}{9}\right)^{1/3} \log_2(e)(n \ln 2)^{1/3}(\ln(n \ln 2))^{2/3} - 14.$$

For elliptic curves and discrete log subgroups, as discussed above, we apply the "half-the-size" principle.

## 6.3 Impact of Special Purpose Hardware

For a (secure) symmetric primitive, the only means to speed up attack time is to use parallelism by running on several general-purpose computers, or, to build a special purpose key-search machine. To our knowledge, the largest such machine built, is the EFF's "Deep Crack", [63], mentioned above. Such hardware is a reality one has to consider, and even without further improvements, a 64-bit key would be recovered in less than a year.

Also for public key (number theoretic) schemes, special purpose cryptanalytic hardware has been proposed.

The TWIRL device and and its (optical) TWINKLE predecessor, [178, 179], have been proposed to speed up the so called sieving-step of factoring algorithms. It seems to be acknowledged (at least by RSA Labs, [100], 2003) that TWIRL would seriously threaten 512-bit RSA keys (which we already know can be attacked by general purpose computers and a quite small effort, [5]) and that the cost of building a TWIRL device that factors 1024-bit RSA keys "reasonably fast" is comparable to the cost of a parallel key-search machine retrieving 80-bit symmetric keys in comparable time.

Bernstein, [19], proposes a massively parallel machine for speeding up the so-called matrix step of factoring algorithms and claims an impact of factoring based keysizes, requiring three times as large keys. Later analysis [116], however, suggests a more moderate 17% increase, and only under "optimistic" assumptions.

Finally, [159] (1999), describes a hardware design estimated to cost $10M that could possibly break elliptic curves over a binary field of size $2^{155}$ in about one month. However, since the design would be less cost-effective for general finite fields, a way to avoid issues is to avoid binary fields. In [207] it is estimated that curves over binary fields should have 8-10 bit larger keys to mitigate the effect of such hardware.

Considering that most of the proposed machines above seem to effect only a limited range of keysizes, or have limited impact, for the purpose of this report, the only special purpose hardware we shall consider (besides hardware for finding symmetric keys) is the machine discussed in [159]. Thus, we propose concerned users to add about 10 bits in the binary field case. However, it is important to keep up to date with advances in special purpose hardware; we by no means rule out future advances in this area.

## 6.4 Quantum Computing

Both of the fundamental intractability assumptions on integer factoring and discrete logarithms break down if a (large) quantum computer could be built as demonstrated by Shor,

[181]. For instance, integers $N$ can be factored in only $O(\log^3 N)$ "steps" on such a machine. We are, however, quite far from realizing such a device. In [193], an experimental result for factoring the "toy" number $N = 15$ is reported with a run-time of just under one second.

For symmetric cryptography, the effect would also be dramatic, though not devastating. By the generic search algorithm due to Grover, [65], key-sizes are in effect cut in half. Also this algorithm has been implemented on small toy examples, [40]. A quantum computer would also imply finding $n$-bit hash function collisions with complexity $2^{n/3}$, [30].

The recommendations in this report assumes (large) quantum computers do not become a reality in the near future.

# Chapter 7

# Recommended Key Sizes

With reference to the above discussion, ECRYPT2 recommends the following minimum key sizes to protect against different attackers. Note that these are *minimum* sizes, giving pro-

Table 7.1: Minimum symmetric key-size in bits for various attackers.

| Attacker | Budget | Hardware | Min security |
|---|---|---|---|
| "Hacker" | 0 | PC | 53 |
| | < $400 | PC(s)/FPGA | 58 |
| | 0 | "Malware" | 62 |
| Small organization | $10k | PC(s)/FPGA | 64 |
| Medium organization | $300k | FPGA/ASIC | 68 |
| Large organization | $10M | FPGA/ASIC | 78 |
| Intelligence agency | $300M | ASIC | 84 |

tection only for a few months.

Given any desired (symmetric key) security level, one may need to convert the symmetric key into an equivalent asymmetric key size. Based on the preceding discussion, we propose the following symmetric/asymmetric size-equivalences, see Table 7.2. We note that it agrees reasonably well with Table 1 of [114].

It may also be interesting to see what today's commonly deployed RSA/DLOG key-sizes offer in terms of equivalent symmetric key-size. This can be found in Table 7.3.

Note that the DLOG and EC recommendations applies for prime order fields. For finite field DLOG schemes, the table may need to be revised taking into account more efficient attacks that are known to exist for DLOG over binary fields. However, we are not aware of any practical deployment of systems based on discrete logarithms in binary fields.

For EC, recommendations for binary fields need to take into account that the field size should be $2^p$, where $p$ is a prime number[1] of bit-size slightly larger than the group/key bit-size. If in addition, the special purpose HW of [159] is considered a threat, roughly an additional 10 bits should be added to the EC key size in this case.

[1] Unless $p$ is a prime, more efficient attacks are known, [205].

Table 7.2: Key-size Equivalence.

| Security (bits) | RSA | DLOG | | EC |
|---|---|---|---|---|
| | | field size | subfield | |
| 48 | 480 | 480 | 96 | 96 |
| 56 | 640 | 640 | 112 | 112 |
| 64 | 816 | 816 | 128 | 128 |
| 80 | 1248 | 1248 | 160 | 160 |
| 112 | 2432 | 2432 | 224 | 224 |
| 128 | 3248 | 3248 | 256 | 256 |
| 160 | 5312 | 5312 | 320 | 320 |
| 192 | 7936 | 7936 | 384 | 384 |
| 256 | 15424 | 15424 | 512 | 512 |

Table 7.3: Effective Key-size of Commonly used RSA/DLOG Keys.

| RSA/DLOG Key | Security (bits) |
|---|---|
| 512 | 50 |
| 768 | 62 |
| 1024 | 73 |
| 1536 | 89 |
| 2048 | 103 |

## 7.1 Recommended Parameters: Non-confidentiality Objectives

The above minimum requirements are intended to keep messages' confidentiality for a certain time. For other security objectives, things may be a little different.

### 7.1.1 Non-repudiation

The possibility to recover security is better here, since one can in principle re-authenticate (and possibly revoke) at regular intervals when one suspects an imminent compromise. Therefore, somewhat shorter keys *may* be acceptable.

### 7.1.2 Message Authentication

Basically, the same recovery mechanisms may be applicable here. Secondly, some applications are of such nature that the value of the data as "authentic" is essentially only a real-time issue, and efficiency/bandwidth may in addition here require a lowered security.

Since there is usually no or little gain (in efficiency) in using short (symmetric) keys, the above recommendations hold (in general, see discussion below) also for authentication key sizes. Assuming a large enough key is chosen (and secure protocols/functions are used), the main threat is therefore forging by "guessing". (We note that there are MACs, e.g. Carter-Wegman based MACs, where one can guarantee that such attacks are the only ones possible.)

For MAC tags, it may be acceptable to use shorter values than the MAC key size. In [70] an absolute minimum tag size of 32 bits is motivated.

An example where even the authentication key may be relatively short is in application of so-called Manual Authentication (MANA) Protocols. A typical application is a one-time authenticated transfer of a data item (e.g. a public key certificate) between two devices using short-range radio. A (one-time) PIN is chosen as a key and is entered (using physical side-channel) in both devices. If implemented correctly, the possibility to forge can be explicitly evaluated, and the key can sometimes be allowed to be shorter than the tag, e.g. a 16-bit key and a 160-bit tag. Note though that these arguments break down if the same key is re-used. See [61] for further details. It should also be noted that such analysis often relies on assumptions such as "physical proximity", which is (partly) used to rule out certain attacks.

In general, when considering minimum acceptable tag size, it is important to analyze whether or not the application is designed in such a way that a possible attacker may have oracle access to a MAC verification mechanism. Short tags should not be used if this is the case.

### 7.1.3 User Authentication

For user authentication (using symmetric key challenge-response type protocols) the importance of the length of the response depends on the time period of validity of the authentication (the "session" length). If an attacker is lucky in guessing one response, he will most likely be detected during re-authentication (if such occurs). The impact of impersonation during a certain time period is of course application dependent, but is generally larger than the impact of faking occasional messages. Responses of 64 bits seem a general minimum. A one-time password (OTP) generator typically produces 24-32 bit values, but is usually used in connection with a longer term, user selected key (a password). Unless physical security of the communication channel is present (security against e.g. hijacking), it is important that the user authentication is coupled with a key agreement followed by integrity protection. That is, at the same time as the response is generated, a session key for integrity protection is generated from the same challenge and used to integrity protect every subsequent message. With proper replay protection of challenges, this limits the effect of a faked response to be essentially a short-lived Denial-of-Service attack.

### 7.1.4 Hash Functions

The main consideration for a secure[2] hash function is the size of the outputs. If the application requires collisions to be difficult to find, the output must be twice the desired security level. This is the case e.g. when used with digital signatures. When used as a keyed hash for message authentication, however, the outputs may often be truncated, see above.

### 7.1.5 Nonces

So-called *nonces* (number used once) are generally required to be as long as the symmetric keys used, i.e. match the security level. This is due to attacks based on the birthday paradox. For example, an $n$-bit nonce (here often called a salt) used for deriving a session key from a

[2]Note the recent developments on MD5 and SHA-1 in Section 10.2.

$k$-bit symmetric key will produce an overall security of about $(n + k)/2$ bits against off-line collision attacks.

## 7.2 Security Levels

Keeping in mind that the security level of Table 7.1 only gives very basic protection (a few months) we need to add some margin to get real protection. At the same time, as we just discussed, there are some applications with special properties that might allow *lower* security requirements, and there may be some very constrained environments where a certain level simply cannot be obtained. We would therefore like to define some *security levels* and quantify what security they reach and in which cases they might be acceptable. That is, rather than mandating use of certain key-sizes, we would like to be more practical and see what security is obtained in different cases.

Table 7.4: Security levels (symmetric equivalent).

| Security Level | Security (bits) | Protection | Comment |
|---|---|---|---|
| 1. | 32 | Attacks in "real-time" by individuals | Only acceptable for auth. tag size |
| 2. | 64 | Very short-term protection against small organizations | Should not be used for confidentiality in new systems |
| 3. | 72 | Short-term protection against medium organizations, medium-term protection against small organizations | |
| 4. | 80 | Very short-term protection against agencies, long-term prot. against small organizations | Smallest general-purpose level, $\leq 4$ years protection (E.g. use of 2-key 3DES, $< 2^{40}$ plaintext/ciphertexts) |
| 5. | 96 | Legacy standard level | 2-key 3DES restricted to $\sim 10^6$ plaintext/ciphertexts, $\approx 10$ years protection |
| 6. | 112 | Medium-term protection | $\approx 20$ years protection (E.g. 3-key 3DES) |
| 7. | 128 | Long-term protection | Good, generic application-indep. recommendation, $\approx 30$ years |
| 8. | 256 | "Foreseeable future" | Good protection against quantum computers |

A few comments. An 80-bit level appears to be the smallest general-purpose level as with current knowledge, it protects against the most reasonable and threatening attack (key-search) scenarios. However, see also further discussion below. The 32 and 64-bit levels should

not be used for confidentiality protection; 32-bit keys offer no confidentiality at all relative to *any* attacker, and 64-bit offers only very poor protection. Nevertheless, there are applications when these levels may be necessary if security is to be provided at all, e.g. for integrity tags. While we certainly do not think this level of security should ever be promoted, it is at the same time important to be aware that some narrow-bandwidth applications would be imposed with a considerable over-head even with such short tags, and it is important to highlight what attacks would then become possible.

The choices 112/128 for the high and very high levels are a bit conservative based on extrapolation of current trends. However, since there exist well-proven standardized components that supports these levels, it still seems to make sense to define them that way.

While both 80 and 128 bit keys provide sufficient security against brute force key-search attacks (on symmetric primitives) by the most reasonable adversaries, it should be noted that the choice between 80 and 128 bits really does matter, in particular if one considers attack models based on pre-computation and large amounts of available storage, i.e. the trade-off attacks of Section 5.1.1. Considering such attacks, 80 bits would be practically breakable, and 128 bits might correspond to an effective 80-bit level, etc. As a simple rule of thumb (in particular considering that the performance penalty is small for symmetric primitives), one may choose to double the key size to mitigate threats from such attacks.

## 7.3 How to Deal with Very Long-term Security

In some applications, e.g. voting, legally binding signatures, protection of state secrets, etc, very high security levels may be needed. As discussed previously, it can be difficult to obtain and maintain such high security levels. Apart from the fact that non-technical issues, outside the scope of this report, tend to matter more, there is also a technical problem appearing. Namely, that is difficult to predict future cryptanalytic advances. As an example, a 1024-bit integer would be factored some 1000 times faster by the Number field sieve, than by the Quadratic sieve (QS) method, so yet unknown future advances could really make a difference. That is, even if extrapolation of key sizes would be a valid approach for the next, say 30 years, one cannot exclude that certain algorithms become more or less completely broken over such long time frames. So, how should we take cryptanalytic progress into account? If cryptanalysis affecting RSA had, over the last, say, 20 years an effect comparable to Moore's law (independently, on top of the actual Moore's law), then it would not be reasonable to assume that that progress stops at this very moment, and it could also be argued to be a stretch of the imagination that it would suddenly go much faster (even though both possibilities are in principle conceivable). For all practical purposes (i.e., a position one can defend) one should assume that future progress follows the trend that we have seen in the past and base one's decisions on that assumption.

Advances have often been done in steps (e.g. the improvement from QS to NFS), and beyond approximately 10 years into the future, the general feeling among ECRYPT2 partners is that recommendations made today should be assigned a rather small confidence level, perhaps in particular for asymmetric primitives.

All is not lost though. By (again) handling security as a matter of due process, it may still be possible to maintain message authenticity and non-repudiation over long periods of time. By keeping up to date on cryptanalytic and technological advances one could, as fear of compromise grows stronger, re-authenticate a message (and the existing signature) with

larger key size and/or different algorithms.

Another approach to message authenticity is to authenticate messages with more than one algorithm. For instance, sign a message with two different algorithms, and consider the (compound) signature valid if and only if both signatures check out. It is here important that keys and parameters are generated randomly and independently, and, depending on what the feared future advances might be, it could be essential that the algorithms are very "different". For instance, signing a message both with RSA and discrete logarithm technology does not offer any additional security if quantum computers become a reality. One can also consider combinations of asymmetric and symmetric keys, e.g. both sign a message, and add symmetric key based integrity.

For confidentiality it may be more difficult to apply the security process thinking since once confidentiality is lost, it is gone forever. However, additional protection may be offered by multiple encryption, again using sufficiently "different" algorithms. The classical one-time-pad scheme may also be applicable to very high security levels, assuming the key management can be solved.

## 7.4 A Final Note: Key Usage Principles

Besides the above recommendations, it is important to be aware of some principles for how to use keys and other parameters.

First, the *randomness* principle must be followed: keys must be randomly or pseudo-randomly generated. If less key-material than needed is available, a pseudo-random function should be applied to obtain the required amount of key. Of course, we do not claim that the attack resistance is still any higher than the size of the input to the random number generator. Rather, we want to stress that other, ad-hoc key-expansion methods could *decrease* the security. As a basis for generating (pseudo)random bits it is crucial to properly seed one's random number generator. This is an issue that is too often overlooked in practical applications.

Secondly, the principle of *key-separation* applies: the same key should never be used for two different purposes (e.g. use distinct keys for encryption and integrity protection, etc). Moreover, the same key should not be used twice with different transforms (e.g. encryption transform A should not use the same key as encryption transform B). For symmetric encryption, do not use the same transform inputs (key, IV, etc) twice to process different messages. For integrity protection, make sure all messages are distinct (e.g. by including a counter) to avoid replay. This recommendation holds also for other primitives, e.g. do not use the same signature key in two different systems.

Finally, for electronic signature schemes, you should not be too "promiscuous" in how you use your secret key to sign. Never sign a document provided by some other party without first applying a secure message padding scheme, including appropriate randomization, see Chapter 14.

# Part II

# Symmetric Primitives

# Chapter 8

# Block Ciphers

## 8.1 Overview

Block ciphers specify keyed, invertible transformations from $b$-bit blocks to $b$-bit blocks, under the influence of $n$-bit keys. That is, a block cipher is defined by two algorithms, $E$, that encrypts, and $D$ that decrypts, such that for all $n$-bit keys $k$, and all $b$-bit blocks $x$, $D(k, E(k, x)) = x$. Most ciphers support only one block-size, but in many cases several key sizes are supported.

A number of security properties are required for a block cipher to be considered secure. Needless to say, one such requirement is that no key-recovery attack (e.g. differential [21], linear attacks [124] etc) of complexity better than $2^n$ is known. Stronger notions of security are for instance various notions of *indistinguishability*. What does that mean? Well, the reader may be familiar with the Turing test for artificial intelligence, where a human can "chat" (via keyboard/screen) with another entity, asking questions like "Describe in single words only the good things that come into mind about your mother", etc. This other entity is either a human or a computer program. The program is considered artificially intelligent, if the user cannot tell if he is chatting with a human or the program. In our case, the attacker gets a plaintext, and a value which is either an encryption of said plaintext (for an unknown, random key), or, a random $b$-bit string. The attacker is challenged to guess which value he got, and to this end he may issue (adaptive) chosen message queries to an "encryption oracle", on any message (except the challenge). If he cannot succeed with probability much better than one-half, the cipher is said to be indistinguishable from a random function. The intuition is that if the encipherments look just like random strings, they should not "leak" useful information about the plaintext. Various relations and equivalences are known between different security notions.

Note that a block cipher should never be used in the raw, but rather run in a specified mode of operation, see Section 8.4. We stress in particular that block ciphers do not provide any real integrity protection, and without added integrity protection, also confidentiality may be lost, [18].

Also the block size may define upper bounds on the security. A small block size may enable creation of dictionaries. Also, "non-random" behavior that might be exploitable starts to appear after about $2^{b/2}$ encrypted blocks. This chapter has therefore been divided according to block size.

For a more extensive discussion of block-cipher security properties, see the NESSIE eval-

uation report, [156]. For more information on explicit block ciphers, see [26].

## 8.2 64-bit Block Ciphers

### 8.2.1 DES

**Definition:** NIST FIPS 46-3, [140].

**Parameters:** 56-bit key and 64-bit block size

**Security:** Key length inadequate.

**Deployment:** Widespread, e.g. RFC 2406 (IPsec), RFC 2246 (TLS).

**Implementation:**

**Public analysis:** Papers such as [21, 124] (see below).

**Known weakness:** Susceptible to differential [21] and linear cryptanalysis [124] and their extensions, which reduces effective key size by roughly 10-12 bits. However the key-size is already acknowledged to be insufficient.

**Comments:** Withdrawn by NIST in 2004.

### 8.2.2 3DES

**Definition:** NIST SP-800-67, [148]. (Also standardized in ISO/IEC 18033-3 [89].)

**Parameters:** 112-bit and 168-bit key and 64-bit block size.

**Security:** Because of the iterative construction of 3DES, there exist key search attacks on 3DES whose work function is significantly less than is suggested by the key length. For three-key 3DES this work function reduces down to $2^{112}$ operations (or even down towards $2^{100}$ under certain attack models) whereas for two-key 3DES the work function reduces from $2^{112}$ down to $2^{120-t}$ if the attacker has access to $2^t$ plaintext/ciphertext pairs ($t > 8$) using the same key.

**Deployment:** Widespread; e.g. 112-bit 3DES widely used in financial applications, 168-bit 3DES featured within IPsec, SSL/TLS, etc.

**Implementation:**

**Public analysis:** Cryptrec report [46].

**Known weakness:** A variety of structural properties are well-known but easily avoided.

**Comments:** In [149], 112-bit 3DES is re-affirmed by NIST only through the year 2010 (on the basis that by the above discussion, it offers 80-bits of security when $2^{40}$ plaintext-ciphertext pairs available to an attacker), and 168-bit 3DES is recommended only through the year 2030 (on the basis that if offers only 112-bits of security).

### 8.2.3 Kasumi

**Definition:** 3GPP TS 35.202, [1].

**Parameters:** 128-bit key and 64-bit block size.

**Security:** As claimed.

**Deployment:** UMTS.

**Implementation:** 3GPP TS 35.203, and 35.204 [2, 3] contain test data.

**Public analysis:** Evaluation report [4], paper [28]. Some provable security relative to linear and differential cryptanalysis has been established [101].

**Known weakness:** In [20] a related-key attack requiring $\sim 2^{54}$ plaintext/ciphertext pairs and complexity $\sim 2^{76}$ was presented. Related-key attacks' practical relevance depends on context and does not impact the 3GPP use.

**Comments:** Variant of MISTY-1. Kasumi to be licensed for use in UMTS.

### 8.2.4 Blowfish

**Definition:** See [177].

**Parameters:** 32–448-bit keys and 64-bit block size.

**Security:** As claimed.

**Deployment:** Popular in IPsec configurations. List of products can be found at [27].

**Implementation:** See [27] (includes test vectors).

**Public analysis:** Vaudenay, [197], found weak keys and known plaintext attacks, but only for round-reduced Blowfish, see also the more recent paper [102]. Rijmen, [168], found a 2nd order differential attack, also against a round-reduced version. These attacks cannot be extended to the full cipher. Schmidt made some observations about the key schedule, [176], that do not seem to effect the security.

**Known weakness:**

**Comments:**

## 8.3 128-bit Block Ciphers

### 8.3.1 AES

**Definition:** NIST FIPS PUB 197, [143]. (Also standardized in ISO/IEC 18033-3 [89], part of Suite-B [157])

**Parameters:** 128-, 192-, and 256-bit key and 128-bit block size.

**Security:** As claimed.

**Deployment:** Widespread, included in TLS, S/MIME, IPsec, IEEE 802.11i, etc.

**Implementation:**

**Public analysis:** NIST report [152], NESSIE and Cryptrec reports [156, 46].

**Known weakness:**

**Comments:** So-called *algebraic attacks* have been put forward as providing potential means to attack AES. While issues remain somewhat opaque, the AES cannot currently be considered vulnerable to such analysis. Instead, security aspects of implementation(s) of the AES may be the most pressing issue. (These aspects are shared with many cryptographic primitives.) So, while not directly related to the cryptographic strength of the AES, cache-timing attacks against unprotected implementations seem to present a practical threat, at least when the AES is not executed in a trusted environment.

Implementation attacks aside, there are still no discernible cryptographic weaknesses in the AES.

For more info on AES, see [10, 53].

While AES is by far the most widely used 128-bit blockcipher, it is by no means the only such algorithm. If a backup algorithm is desired, one might for instance consider Camelia, [89].

## 8.4 Modes of Operation

To actually perform encryption on large messages, a block cipher needs to be run in a *mode of operation*. Choosing a secure mode is important as the way in which the block cipher is used could lead to insecurity, even if the block cipher as such is secure.

Numerous modes with many different properties exist and have been proposed. While some modes also provide for message integrity, notice that this is not the case for any of the three modes below. If integrity is desired, a secure MAC must be used together with these modes, see Chapter 11. Integrity preserving modes can be found in e.g. the NIST specification [147]. A comprehenshive overview of authenticated encryption can also be found in [23].

Below we discuss only the three most common modes, their properties and recommendations for how to apply them securely. Note that different standards for modes of operation may not be interoperable, e.g. due to different padding and initialization (IV) schemes. An example of one standards specification is ISO/IEC 10116, [86]. Another set is that by NIST, [144, 146].

As noted below, some modes have formally been proven to be secure in certain models, assuming the block cipher used is secure. Generally speaking, the quantitative measure of security obtained by these proofs decay with the number of message blocks processed under a given key. Security, in a strong sense, is lost as the number of processed blocks approaches $2^{b/2}$ for a $b$-bit block cipher. Thus, re-keying should occur well before this bound is reached. In practice, this is only an issue for the 64-bit block ciphers above.

IVs should follow similar size considerations as for nonces, Section 7.1.5.

### 8.4.1 Electronic Code Book Mode (ECB)

This mode should only be used to encrypt messages that are smaller than or equal to the block size, as information about the message otherwise leaks (e.g. it is possible to tell if two parts of the messages are identical or not). For the same reason, a given key should only be used to encrypt one single message. For messages smaller than the block size, message padding to fill an entire block is necessary, thus creating (slight) bandwidth overhead. If an error in the cipher text occurs, the decryption will produce "garbage" due to error propagation, i.e. about half of the bits of the plaintext will be changed.

### 8.4.2 Cipher Block Chaining (CBC)

CBC is probably the most widely used mode of operation. Besides the key, an initialization vector (IV) is needed. This IV should be random and independent for each message. The same key/IV pair should not be used to process more than one message. Integrity of the IV should be assured, as it will otherwise enable an attacker to introduce predictable changes in the first decrypted plaintext message on the receiver side and/or perform certain attacks.

CBC either requires padding, or a special treatment of the last block, e.g. applying so called OFB mode to the last block or using so-called cipher text stealing. Both of these may however leak information and/or are sensitive to certain attacks, [139]. Note that padding schemes are generally sensitive to side-channel attacks, where the attacker can inject messages towards the receiver and observe padding-error notifications returned, [196, 138]. This is yet another reason why integrity/authenticity is important: without it, confidentiality may also be lost.

If an encrypted block is subject to bit-errors, the corresponding plaintext block will be garbled, and the following block will have bit-errors matching those of the erroneous cipher text block.

If IVs are explicitly signaled between sender and receiver (or otherwise agreed upon) for each message, CBC has a random access property in that the order of arrival of messages (e.g. IP packets) does not matter. However, within a given message, processing must be done blockwise sequentially.

If the block cipher is secure, CBC can be proven to be secure, [15].

### 8.4.3 Counter Mode (CTR)

CTR basically produces a stream cipher (see next chapter) from a block cipher. Besides the key, an IV is needed. The IV should be unpredictable to provide security, e.g. against attacks such as [131]. It is of paramount importance that the same IV/key pair is not used to protect two different messages.

CTR can provide random access also within a given message and does not require padding.

As we have argued, modes of operation will not in general provide any message integrity. This is particularly true for CTR as an attacker trivially can modify bits of the plaintext.

If the block cipher is secure, CTR can be proven to be secure, [15].

### 8.4.4 Hybrid Encryption

Later in this report we define schemes for *Hybrid Encryption* (Key/Data Encapsulation Modes, KEMs/DEMs), see e.g. Section 16.2.1. These combine asymmetric and symmetric

cryptography. Hybrid schemes can sometimes be proven secure, but requires careful choice of the symmetric primitives (block ciphers/modes, MACs, etc). It is recommended to use only primitives allowed/defined by the respective standard. For instance, the standard ISO/IEC 18033-2 [88] specifies a DEM using CBC (ISO/IEC 10116) and a MAC according to ISO/IEC 9797-2. On the other hand ISO/IEC 19772 standardises six authenticated encryption techniques, and was published in February 2009. The techniques it covers are: OCB 2.0, key wrap, CCM, EAX, encrypt-then-MAC (combining any symmetric encryption method with any MAC) and GCM.

# Chapter 9

# Stream Ciphers

## 9.1 Overview

A stream cipher is an algorithm that takes a key (and possibly other information) and produces a (for most practical purposes) arbitrary long sequence, the *keystream*. This keystream is then combined with the cleartext to form the cipher text. In most commonly used stream cipher applications today, the keystream is bit-wise added modulo 2 to the cleartext (XORed), this is then called a *binary additive stream cipher*. Some ciphers use another combination operation, which in some cases can even provide some integrity protection. Note that it is obvious that a binary additive stream cipher in itself provides no integrity protection; an attacker can flip individual bits (even if he does not know if he flips one to zero or vice versa). Binary additive stream ciphers *should* therefore always be used with integrity protection.

Stream ciphers come in two flavors. *Synchronous* stream ciphers require sender and receiver to maintain synchronization within the key stream by external means (e.g. including an explicit synch value in each message). *Self-synchronizing* stream ciphers, allow temporary loss of synch, which is then automatically regained after a while. Essentially all practically used stream ciphers today are synchronous.

It is obvious from construction that the security of a stream cipher depends on the "randomness" of the keystream, and if the stream appears indistinguishable from true random bits, security follows immediately. However, few stream ciphers have managed to resist all distinguishability attacks, but if a distinguishing attack requiring huge amounts of known keystream exists, it may not be a disaster for the practical security.

Why would one choose a stream cipher rather than a block cipher? Usually, any or all of the following three properties may be motivation:

1. Stream ciphers are usually designed with high speed and/or constrained computing environments as requirements, hence many stream ciphers are quite fast and "lightweight" (though note that some constructions have historically at the same time suffered security-wise).

2. Stream ciphers do not require padding of messages, hence they may offer bandwidth savings in critical applications.

3. Stream ciphers do not propagate bit-errors; single transmission bit-errors in ciphertext translates to single bit-errors (in the same bit positions) after decryption.

The last motivation is probably also the most controversial since, as noted, it also means that attackers can flip message bits if integrity protection is not used, and integrity protection would at the same time work against the error-resistance of the application. Nevertheless, many voice-over-radio systems (without integrity protection) use stream ciphers since the voice decoders usually have high tolerance to single bit-errors, but cannot cope well with frame-errors.

Note that binary additive stream ciphers may lose all security if the same key (or more precisely, the same keystream) is used to encrypt two different messages (creating so-called two-time pad).

For a more extensive discussion of stream-cipher security properties, see the NESSIE evaluation report, [156].

### 9.1.1 A Note on Pseudo random Number Generation

In principle, any stream cipher can function as a pseudo random number generator (PRNG). Are there any differences in requirements on a PRNG and a stream cipher? Looking purely at the requirements on the outputs, using a strong definition of stream cipher security, there is no real difference, both need to produce outputs computationally indistinguishable from random. In practice there are usually some differences in requirements though:

- Stream ciphers are usually called for due to bandwidth saving (avoid padding) or processing speed, otherwise one can just use a block-cipher in a suitable mode (which could be a stream cipher emulating mode). A PRNG can on the other hand often be acceptable even if the speed is quite low.
- As mentioned, a distinguishing attack on a stream cipher may not be catastrophic, but if a PRNG is used to generate keys for other ciphers, it could be a completely different story. There *may* thus be reason to have stronger security requirements on a PRNG.
- Re-keying of a stream cipher is usually done to avoid "wearing out" the key or to achieve synchronization, and the requirements on avoiding key-stream re-use are quite well understood. A PRNG may be "re-seeded" to improve the randomness of the internal state, i.e. to "add entropy" to an existing state, and this is currently done more as an art than a science.

### 9.1.2 eStream

As can be seen below, very few stream ciphers are included in this report, mainly due to the lack of publicly available stream ciphers that meet the above requirements while maintaining sufficient security. However, ECRYPT in 2008 finalized a public development effort of efficient, secure stream ciphers, [55]. The following stream ciphers are included in the portfolio.

The hardware ciphers all support 80-bit keys and aim for an 80-bit security level. While Salsa20 claims 256-bit security when used with 256-bit keys, all the other software ciphers claim 128-bit security. For more information, we refer to [55].

#### F-FCSR-H

After finalizing the eStream effort, a severe attack on F-FCSR-H appeared. Thus, the stream cipher F-FCSR-H was removed from the portfolio.

Table 9.1: eStream Portfolio.

| Software-optimized | Hardware-optimized |
|---|---|
| HC-128 | Trivium |
| Rabbit | Grain v1 |
| Salsa20/12 | MICKEY v2 |
| SOSEMANUK | |

## 9.2 RC4

**Definition:** see [80, 81].

**Parameters:** variable key size.

**Security:** While no key recovery attack is known when used directly as a keystream generator, RC4 is highly sensitive to attacks which exploit re-keying and re-initialization.

**Deployment:** Widespread, e.g. SSL/TLS, IEEE 802.11b, etc.

**Implementation:**

**Public analysis:** Cryptrec [46], see also e.g. [80].

**Known weakness:** Various distinguishing attacks apply, e.g. [122]. Some key recovery attacks are known for specific implementations with re-keying, [22]. State recovery attacks have also been found [130]. The first bytes of generated keystream are particularly vulnerable to cryptanalysis. The best key recovery attack on the implementation WEP is a passive one [198].

**Comments:** Recommendations often include dropping the first 512 bytes of generated keystream. However, due to the ease of misuse, ECRYPT2 recommends against using RC4 as a general purpose stream cipher.

## 9.3 SNOW 2.0

**Definition:** [90]

**Parameters:** 128 and 256-bit keys.

**Security:** No attacks of any practical relevance are known.

**Deployment:** Used in DisplayPort, [199].

**Implementation:**

**Public analysis:** NESSIE [155]. See also [204, 129, 158].

**Known weakness:** Given about $2^{174}$ bits of keystream and $2^{174}$ work, it is possible distinguish SNOW 2.0 outputs from true randomness, see [158]. The attack is non-trivial only for 256-bit keys and the amount of keystream needed will not occur in practice.

**Comments:** SNOW 2.0 is an enhancement of "SNOW" as submitted to NESSIE. SNOW 2.0, in turn, exists in a further modified version, "SNOW 3G", and has been adopted by ETSI SAGE for inclusion in the 3GPP UMTS standard. The main difference in SNOW 3G is the addition of a second S-box giving higher resistance against possible future advances in algebraic cryptanalysis.

# Chapter 10

# Hash Functions

## 10.1 Overview

Cryptographic hash functions are widely used in computer and network security applications. They operate on a message of (for practical purposes) arbitrary size and produce a fixed-size fingerprint, or digest, of the message. Depending on the application, some or all of a number of security properties are required from the hash function $h(\cdot)$. Potential properties that we might appeal to are:

**Pre-image resistance:** Given an output $y = h(x)$ (but not a corresponding input $x$) it is practically infeasible to find $x$. Such a property might be useful when one wishes to commit to the value $x$ at some point in time while keeping the value of $x$ secret until later.

**2nd pre-image resistance:** Given an output $y = h(x)$ *and* a corresponding input $x$ it is practically infeasible to find another input $z \neq x$ such that $h(z) = h(x)$. This property might be used to prevent some party from changing from a committed value.

**Collision resistance:** It is practically infeasible to find *any* pair of distinct inputs $x$ and $z$ such that $h(x) = h(z)$. This property is often required to protect electronic signature schemes against forgeries. In such schemes the hash of a message is typically signed as a representation of that message. Thus if an attacker can find two inputs $x$ and $z$ that collide with respect to some hash function, then the attacker might be able to re-use a legitimate signature on $x$ as a correct, but falsely-obtained, signature on $z$.

**Random oracle property:** The function $h(\cdot)$ "behaves" as a randomly chosen function. Assuming this property holds sometimes makes it possible to formally prove the security of public key encryption and signature schemes. We shall discuss this more in Section 12.1.

While one can construct contrived examples of functions that are collision resistant but not pre-image resistant, the hash function properties have been ordered in terms of the difficulty faced by an opponent, with the task of finding a pre-image being the hardest. In the absence of any analytic weaknesses, only brute force methods are available to the attacker. More precisely, if $n$ is the size of the hash outputs, one would from a secure hash expect around $2^n$ operations to be required to break the first two properties (though this decreases as the

number of available targets increase) and around $2^{n/2}$ operations to break the property of collision resistance (due to the birthday paradox). Consequently, one needs to choose a secure $h$ with a large enough $n$ so that these numbers meet application-dependent requirements on "practical infeasibility".

It is quite clear from construction that none of the hash functions here can be considered "random oracles" in a generic sense, and it has in fact been established that no fixed function can have all required properties, [34]. The practical implications of the so-called random oracle model has therefore been debated, and we shall return to this in the asymmetric algorithm part of this report.

For a more extensive discussion of hash function security properties, see the NESSIE evaluation report, [156]. For more information on explicit hash functions, see [54, 74].

## 10.2 Recent developments

A lot of advances in analysis of iterated hash functions, in particular from the MD5/SHA family, has been made since work on the first revision of this report started, e.g. [6, 96, 97, 103, 104, 191, 190, 201, 202, 203].

To summarize, MD5 has to be considered completely broken from collision resistance point of view (explicit collisions can be demonstrated), and SHA-1 provides only a very marginal security. In both cases (but with different complexity) collisions can be found from *known* IVs. This means that applications to message authentication (which are based on secret IVs) are not directly threatened, though some concerns about further advances may be raised, see Section 11.2. Also, it is not possible to completely choose the messages that collide, and one may argue that also signatures therefore are (in practice) secure as long as only "random" collisions can be found. The latter is, however, a too optimistic assumption. It has been shown that both syntactically correct public key certificates, as well as pairs of human readable colliding documents can in practice be produced, [36, 47, 118, 190]. Therefore, use of SHA-1 and in particular MD5 should be avoided with signatures. A more fundamental solution to strengthen digital signature schemes relying on not-collision resistant hash functions would be randomized hashing [68]. See also [52] for more discussion.

Note that some "obvious" techniques aiming at improving/repairing the security of some hash function(s) may not have the effect one hopes. For instance, simple schemes such as concatenating the output of some commonly used hashes may not help, see [96].

## 10.3 MD5

**Definition:** RFC 1321, [170].

**Parameters:** 128-bit hash output, max input size $2^{64} - 1$ bits.

**Security:** Not collision-resistant. Indeed collisions can be found within seconds on a common PC.

**Deployment:** Widespread; e.g. in SSL/TLS, IPsec, etc.

**Implementation:** C-source code available in RFC 1321, [170].

**Public analysis:**

**Known weakness:** Collisions have been found, [191], with low computational complexity when the form of the message is restricted over 596 bits (though the total message length may be greater and the form of the remainder is unrestricted). Collisions for (valid) public key certificates have been reported, [119, 190], and password recovery attacks applicable to MD5 usage in concrete applications are known, [120, 175]. A practical attack creating a rogue Certification Authority has been demonstrated, [191]. A preimage attack with a complexity of $2^{124.4}$ is also known [174]. Further cryptanalytic improvements should be anticipated.

**Comments:** MD5 should not be used in new deployments and should be phased-out of existing applications (in particular signatures) as soon as possible. Further comments are available via the ECRYPT statement on hash functions, [52].

## 10.4 RIPEMD-128

**Definition:** see [169].

**Parameters:** 128-bit hash output, max input size $2^{64} - 1$ bits.

**Security:** As claimed; collision search requires $2^{64}$ iterations of the compression function. However, this is no longer adequate.

**Deployment:** Unknown.

**Implementation:** see [169].

**Public analysis:**

**Known weakness:**

**Comments:** While collisions on RIPEMD are reported in [201], RIPEMD is a significantly different design to RIPEMD-128. However, collisions for reduced, 3-round, RIPEMD-128 are reported in [133].

## 10.5 RIPEMD-160

**Definition:** ISO/IEC 10118-3, [87] (see also [169]).

**Parameters:** 160-bit hash output, max input size $2^{64} - 1$ bits.

**Security:** As claimed; collision search requires $2^{80}$ iterations of the compression function.

**Deployment:** Permissible algorithm in IPsec, IEEE Std 1363, and OpenPGP.

**Implementation:** see [169].

**Public analysis:** Cryptrec report [46].

**Known weakness:**

**Comments:** While collisions on RIPEMD have been reported, RIPEMD is a significantly different design to RIPEMD-160. Partial attacks on 2- or 3-round reduced-versions (out of 5 rounds) should be anticipated, [134].

## 10.6 SHA-1

**Definition:** NIST FIPS 180-1 and NIST FIPS 180-2, [141]. Also included in IEEE Std 1363, ISO/IEC 10118-3, etc

**Parameters:** 160-bit hash output, max input size $2^{64} - 1$ bits.

**Security:** Not collision resistant. Full collisions have not yet been found, but may be expected at any moment.

**Deployment:** Widespread (included in e.g. IKE/IPsec).

**Implementation:** RFC 3174.

**Public analysis:** Cryptrec report [46].

**Known weakness:**

**Comments:** Collisions on SHA-1 can be found using $2^{69}$ operations, [202], and newer results even indicates somewhat lower complexity, [135, 121, 203]. Explicit collisions for the full SHA-1 have (yet) not been found. The current "record" for an explicit collision is when SHA-1 is reduced from 80 to 70 rounds, [38]. Preimage attacks for up to 45 rounds have been reported [39].

We recommend against using SHA-1 in new applications, and signature applications with medium to high security should as soon as possible phase out use of SHA-1. Use in message authentication, e.g. HMAC, does not appear immediately threatened, though some caution could be motivated, see Section 11.2.

## 10.7 SHA-224, SHA-256

**Definition:** NIST FIPS 180-2, [141] (Also part of Suite-B [157], ISO/IES 10118-3).

**Parameters:** 224-bit and 256-bit hash outputs respectively, max input size $2^{64} - 1$ bits.

**Security:** As claimed; collision search requires $2^{112}$ and $2^{128}$ iterations of the compression function respectively.

**Deployment:** Likely to become widespread.

**Implementation:**

**Public analysis:** Cryptrec report [46]. See also [62, 73].

**Known weakness:**

**Comments:** Collisions on SHA have been reported. While SHA has some similarities, it is also a significantly different design to SHA-224 and SHA-256. SHA-224 is identical to SHA-256, except that it uses a different IV and truncates the output. Simplified variants of SHA-256 have been analyzed in [125, 133, 210].

Practical collision attacks for up to 24 (out of 64) steps have been reported [83].

## 10.8 SHA-384, SHA-512

**Definition:** NIST FIPS 180-2, [141] (Also part of Suite-B [157]).

**Parameters:** 384-bit and 512-bit hash outputs respectively, max input size $2^{128} - 1$ bits.

**Security:** As claimed; collision search requires $2^{192}$ and $2^{256}$ iterations of the compression function respectively.

**Deployment:** Unclear. Included in e.g. ISO 10118-3.

**Implementation:**

**Public analysis:** Cryptrec report [46].

**Known weakness:**

**Comments:** Collisions on SHA have been reported. While SHA has some similarities, it is also a significantly different design to SHA-384 and SHA-512. SHA-384 is identical to SHA-512, except that it uses a different IV and truncates the output.

Collision attacks for reduced variants of both SHA-512 and SHA-384 up to 24 (out of 80) steps have been reported [83].

## 10.9 Whirlpool

**Definition:** ISO/IEC 10118-3, [87] (see also [82]).

**Parameters:** 512-bit hash output, max input size $2^{256} - 1$ bits.

**Security:** As claimed; collision search requires $2^{256}$ iterations of the compression function.

**Deployment:** Unclear.

**Implementation:** see [82].

**Public analysis:** NESSIE, [155].

**Known weakness:** A collision attack for Whirlpool reduced to 4.5 rounds (out of 10) with a complexity of $2^{120}$ has been reported in [136].

**Comments:** Built using AES-like components and has a different design philosophy to the MD-family.

# Chapter 11

# Message Authentication Codes

## 11.1 Overview

MACs aim to provide integrity protection. Given a key, $k$, they operate on a message, $M$, by computing an $m$-bit check-value $MAC(k, M)$, which is then usually appended to $M$ before transmission/storage. The receiver/retriever of the message similarly computes the "expected" MAC value and compares it to the presented one.

The basic security notion for MACs is that without knowledge of the key, it should be infeasible for an attacker (even after seeing many, possibly even chosen, $M, MAC(k, M)$-pairs) to produce a new, $(M', t')$ pair so that $t' = MAC(k, M')$, other than by pure chance, guessing the value with probability $2^{-m}$. Attacks included in this notion are:

**Key recovery:** retrieve the $n$-bit key, $k$.

**Insertion:** create $(M, t)$ such that $t = MAC(k, M)$.

**Modification:** observing some $(M, t) = (M, MAC(k, M))$ try to modify it into $(M', t')$ so that $t' = MAC(k, M')$.

Note that the first attack is *verifiable*, whereas the two other usually are not: the attacker knows when he has found the right key, but may not be able to tell if a candidate tag value is correct. So, both $n$ and $m$ are upper bounds on the security with respect to different attacks, but a smaller $m$ can usually be accepted since non-verifiability of "tag-guessing" would only create occasional forgeries, and/or, since many applications have a natural built-in re-try limit.

Often, integrity is provided in addition to confidentiality, and one often sees discussion on whether to use an encrypt-then-authenticate, or, an authenticate-then-encrypt strategy. From security point of view, the best way is to first encrypt, then authenticate the encrypted value. First, this has the benefit that the receiver can authenticate before spending resources decrypting. Secondly, it seems intuitively clear that adding integrity check values (a form of "redundancy") before encrypting, cannot improve confidentiality, and will (at least in theory) actually reduce security.

For a more extensive discussion of MAC security properties, see the NESSIE evaluation report, [156]. Annex B of ISO/IEC 9797-1, [84], also provides a nice summary for the algorithms included in that standard.

## 11.2 HMAC

**Definition:** RFC 2104 [112].

**Parameters:** A key and a hash function, often MD5 or SHA1. HMAC-MD5 provides $m$-bit tag for $0 \leq m \leq 128$ HMAC-SHA-1 provides $m$-bit tag for $0 \leq m \leq 160$. Key size depends on the hash function and the standard.

**Security:** As for all iterated MAC constructions, the security is not only limited by the size of the MAC tag, but also by the square root of the size of the internal state (due to birthday attacks). For HMAC-MD5 this results in a security level of $2^{64}$ and for HMAC-SHA-1 a security level of $2^{80}$. The HMAC construction has provable security under certain assumptions on the hash function, [14, 13].

**Deployment:** Widespread: SSL/TLS, IPsec, etc. Included in ISO 9797-2, NIST FIPS 198.

**Implementation:**

**Public analysis:** NESSIE report, [156].

**Known weakness:**

**Comments:** Note that increasing the key length beyond the hash function input block size (512 bits for MD5 and SHA1) does not offer any increase in security. Some standards (e.g. IPsec) allow the hash output to be truncated. Security proofs for HMAC depend upon plausible (but untested) properties of the hash functions covered elsewhere in this algorithm report. The recent advances in the cryptanalysis of MD5 (see Section 10.3), and specifically HMAC-MD5 (e.g. [41, 106, 166, 58, 200]), suggest that implementers move away from HMAC-MD5 as soon as possible.

**Note:** Please also refer to Section 10.6 for latest developments surrounding collision resistance of SHA-1. Together with recent progress on analyzing HMAC-SHA1 with round-reduced SHA-1, [41, 106, 166, 167], caution needs to be considered for the use of HMAC-SHA1.

## 11.3 CBC-MAC-X9.19

**Definition:** ANSI X9.19 [7] (a.k.a. ANSI Retail MAC). Also included in ISO/IEC 9797-1.

**Parameters:** 112-bit key and 64-bit MAC output (with optional truncation)

**Security:** Beyond $2^{32}$ MAC operations using the same key, security breaks down allowing MAC forgery and efficient key recovery attacks, [165].

**Deployment:** Widespread

**Implementation:**

**Public analysis:** e.g. [165] (see above).

**Known weakness:** A wide-range of attacks requiring $2^{32}$ MAC operations are known.

**Comments:** Implementers are recommended to move to alternative schemes for future applications unless frequent re-keying is used.

## 11.4 CBC-MAC-EMAC

**Definition:** ISO/IEC 9797-1, [84].

**Parameters:** Block cipher dependent, but particularly suited to the AES. Key length matches AES key lengths and provides an $m$-bit tag for $0 \leq m \leq 128$. 64 bits is recommended.

**Security:** For MAC-forgery the birthday bound gives a work effort of $2^{64}$ operations and for key recovery the work effort is $2^k$ operations where $k$ is the length of the user-supplied key.

**Deployment:** Potentially widespread.

**Implementation:**

**Public analysis:**

**Known weakness:**

**Comments:**

## 11.5 CMAC

**Definition:** NIST SP800-38B, [145]. Soon to be included in ISO/IEC 979-1

**Parameters:** A key and a block cipher, typically AES. The tag can have any length less than or equal to the block size of the block cipher, although a tag length of at least 64 is normally recommended. The message can have any length.

**Security:** If a secure cipher with block length $m$ is used, and no more than $b$ blocks are authenticated under a given secret key, then the probability of successful forgery is bounded by $b^2/2^{m-2}$. For example, with 128-bit AES, if no more than $2^{43}$ blocks are authenticated — whether $2^{40}$ messages of 8 blocks or 8 messages of $2^{40}$ blocks — then the probability of successful forgery is bounded by $2^{86}/2^{126} = 2^{-40}$.

**Deployment:** Becoming widespread — the default choice today for an AES-based MAC. Included in IPsec, [186].

**Implementation:**

**Public analysis:**

**Known weakness:**

**Comments:** Security fails completely (all subkeys are revealed) if the result of encrypting the all zeroes string under the CMAC secret key is revealed. Handschuh and Preneel, in the full version of [71], note that the encryption of the all zeroes string is sometimes used in the banking industry to check that a secret key has been correctly shared. Iwata [92] has pointed out a much more serious problems, namely that having such a key check value is also devastating for CBC-MAC without final encryption (as allowed by ISO standards).

# Part III

# Asymmetric Primitives

# Chapter 12

# Mathematical Background

Below we give a brief overview of the mathematics behind asymmetric techniques, a more extensive discussion can be found in [156].

Asymmetric schemes are based on the assumed intractability of some mathematical problem—one hopes that "breaking" the scheme is essentially as hard as solving a (presumed) difficult mathematical problem. This somewhat contrasts symmetric schemes. Though mathematical theory exists for building-blocks of symmetric primitives, the construction of e.g. a block cipher is still more of an art than a science, relying on experience from known attacks, trying to put together building-blocks so as to avoid them. Are asymmetric schemes more secure? This is an often debated question, there is really nothing that guarantees this, since the security at the very bottom still rests on an unproven assumption. Nevertheless, there is a certain attractiveness in basing the security on some quite well-studied mathematical problem that nobody (often despite large efforts) has been able to solve.

## 12.1 Provable Security

In some cases, one can say a little more than just that the security is "based" on a mathematical problem. It is sometimes possible to formally prove that the security of an asymmetric scheme is "as high as" the difficulty of solving an underlying mathematical problem. What does this mean? Assume that the mathematical problem is denoted $M$ (this may be the problem of integer factorization) and that the asymmetric scheme is denoted $A$. A proof could for instance (intuitively, and simplified) say that:

> *if* scheme $A$ can be broken with computational effort $T$, then problem instances of $M$ can be solved with computational effort $f(T)$,

for some function $f$. Now, if $f(T)$ is less than the presumed (state-of-the-art) difficulty of solving instances of $M$, this would mean that an attack on $A$ would be an (seemingly unlikely, or at least unexpected) break-through in computational theory, which gives some confidence in $A$'s security. We here have what we in computational complexity theory call a *reduction from solving $M$ to breaking $A$*. Things are, however, slightly more complex and there are two issues we would like to point out.

First, the statement of the exemplified reduction above is made in the so-called *standard model*. That is, no further assumptions are made/needed, the statement/security proof is self-contained. In practice, such proofs may be difficult to find. There have therefore been proposed alternative *proof models* to the standard model. We here mention two.

**The Random Oracle Model (ROM):** In this model, some component of the asymmetric primitive, often a hash function, in assumed to behave precisely as a completely random function; a random oracle.

**The Generic Group Model:** In this model, one assumes that the group in which the arithmetic of the asymmetric scheme is carried out has no "exploitable" properties. E.g. the elements of the group can only be treated as abstract objects, on which one can perform arithmetic, but one cannot exploit things such as representation of group elements, etc. Some elliptic curve schemes can be proven secure in this model.

Either of these assumptions clearly makes the proof weaker in the sense that it then rests on even more unproven assumptions. Moreover, a main question is whether these models have anything to do with reality. In some cases they at least do not appear unreasonable, in other cases, simple counterexamples can be given, showing that these properties do not hold. In particular, concerning ROM, it is known that there are (contrived, but still) protocols that *are* secure when implemented with a "real random oracle", yet they become insecure no matter what concrete hash function one uses, [34]. Another way of reasoning is that a proof in any of these extended models at least is an indication of some soundness of the scheme. We will not debate this issue in more depth, we only remark that clearly, a proof in the standard model would be qualitatively preferred.

Besides qualitative properties of the proofs, there is also a second, quantitative aspect. In the example above, it was stated that "If scheme $A$ can be broken with computational effort $T$, then problem $M$ can be solved with computational effort $f(T)$". An important question is how $f(T)$ grows as a function of $T$. This is usually referred to as the *tightness of the reduction from $M$ to $A$*. We prefer a function $f$ which does not grow too fast, say e.g. $f(T) = T^2$, or even linearly; $f(T) = 7T$. Why is this? First of all, if $f$ grew too fast, $f(T)$ might be larger than the state-of-the-art method to solve $M$, and the proof would say nothing[1]. But one can make another, qualitatively more accurate interpretation of the reduction as

> $A$ is at most a little easier to break than problem $M$ is to solve.

The amount of "little easier" is what is determined by $f$, and hence by the tightness of the reduction. The tighter the reduction, the stronger the confidence in the security of $A$. We will informally in the sequel just refer to reductions as *tight* or *loose*. Exact details can be found in [156], or referenced papers.

Finally, we note that sometimes, the problem $M$ could also be the problem of breaking a(nother) cryptosystem, i.e. one then shows that $A$ is essentially as secure as $M$.

A discussion on some existing misconceptions about these aspects of provable security can be found in [108].

## 12.2 Choice of Primitive

The primitives in this report (and their security proofs, where such exist) are based on reductions from the RSA/factorization problem or from the discrete logarithm problem in a suitable group. To briefly recapitulate, the factoring assumption (for our purposes) states that given $N = pq$, for primes $p, q$ of roughly equal size, it is "hard" to find the individual

[1] We have cheated by suppressing the fact that the size of the problem instances of $M$ (which determines complexity of $M$) typically depends on the key size, $n$, of $A$.

factors $p, q$. The (possibly strictly) stronger RSA assumption states that it is hard to invert the RSA function $x^e \bmod N$, $e \geq 3$. Finally, the discrete logarithm assumption states that given $g^x$ (in a suitable group), it is "hard" to find $x$. A number of additional options and considerations also arise when selecting asymmetric primitives.

For factoring-based schemes, there are two main choices. First, the basic primitive: RSA or Rabin. Rabin, which is basically RSA with fixed exponent $e = 2$ is equivalent to factoring, but general RSA is not known to be. This can be viewed as an advantage for Rabin, but it also implies that being able to break confidentiality (or forge signatures) of one message is equivalent to finding the secret key.

For discrete-log based schemes, there are two main choices. First, the choice of group (either the multiplicative group modulo a prime or an elliptic curve) and whether the group is part of the public key or is it common to all users of the scheme. Secondly, the choice of the actual scheme where different security proofs are known for different schemes, more on this below. Some DLOG based schemes actually rely on the *Computational Diffie-Hellman assumption* (CDH): given $g^x, g^y$, it is "hard" to find $g^{xy}$, which is no harder than computing discrete logarithms. Yet other schemes rely on the (possibly strictly) stronger *Decisional Diffie-Hellman assumption* (DDH): given $g^x, g^y, g^z$, it is "hard" to tell if $z = xy$ or not. Some relations between the hardness of these problems are studied in [126, 127, 128].

### 12.2.1 Other types of Primitives

Pairing based schemes (based on variations of the DLOG problem, e.g. groups where there is an assumed "gap" between CDH and DDH), have some special features/advantages (e.g. possibility of "certificate-less", identity based cryptography) and could also be considered. No such scheme is however included due to lack of deployment and/or maturity. In particular, recommendations for complete sets of parameters, giving a certain "symmetric key-size equivalence" is still somewhat difficult to make with sufficient assurance/confidence.

There are also some proposed schemes, whose security depend on the difficulty of certain lattice problems, e.g. NTRU, [77]. NTRU provides advantages in terms of performance, but similar caveats regarding parameter choices as for pairing-based schemes apply.

Common to both the pairing based schemes and NTRU is also that there is yet no widespread deployment and standardization is only just finishing (e.g. IEEE P1363.1).

## 12.3 Non-cryptographic Attacks

As with most public key schemes, various side-channel attacks may need to be considered, depending on the deployment environment, see e.g. [156].

# Chapter 13

# Public-key Encryption

## 13.1 Overview

Somewhat simplified, an asymmetric encryption scheme consist of three algorithms $(G, E, D)$. Here, $G$ is the key generation algorithm that generates private/public key pairs $(e, d)$. Anybody knowing $e$ can encrypt messages $c = E_e(m)$ so that only the party knowing the corresponding $d$ can retrieve $m = D_d(c)$. Since no deterministic, public key scheme can be secure in a strong sense, in practice the situation is a bit more complex, requiring random padding of messages, etc.

A number of options and considerations arise when selecting asymmetric encryption primitive. As usual, a main choice is whether to select a factoring based or a DLOG based schemes. While other primitives are also possible, see Section 12.2.1 above, for the time being only these two types are treated here.

For factoring-based schemes, there are two main choices. First, the basic primitive: RSA or Rabin. Besides what we have already discussed, Rabin with $e = 2$ is faster than RSA with $e \geq 3$, but Rabin needs some redundancy to be sure that the decrypted message is the cleartext. The current revision of this report does not include any Rabin based scheme due to the lack of wide-spread deployment. Next, one needs to select a padding scheme. For RSA, PKCS#1v1.5 has no security proof based on a reduction to a widely studied problem. OAEP, included in the new PKCS#1v2.1 does, but relies on the ROM and the reduction of the proof is loose.

The present revision of the report does not include any general purpose DLOG based encryption scheme, mainly due to lack of deployment.

### 13.1.1 Security Notions

A number of security notions have been proposed for asymmetric encryption schemes. Today, a strong and probably the most widely accepted measure is that of *indistinguishability under adaptive chosen ciphertext attack*. Informally, the attacker first gets access to a decryption oracle, which decrypts chosen ciphertexts on request. Next, the attacker gets a challenge: a pair of messages, and a value which is the encryption of one of them, with the task of telling which message it corresponds to. To this end, the attacker may issue additional decryption queries (except on the challenge ciphertext), and is considered successful if it can decide which message it got with non-trivial probability (slightly better than 1/2). This notion of security is abbreviated IND-CCA2, and unless otherwise noted, is what we below mean by "secure".

Again, for a more extensive discussion of the security of asymmetric schemes, we refer to the NESSIE evaluation report, [156].

### 13.1.2 Consideration: Hybrid Encryption

The amount of data that can be efficiently protected by the public key techniques below is rather limited, and it is usually a good idea to use hybrid encryption where a key is transported by public key means, and said key is then used with symmetric techniques to protect the actual data. Special public key based key encapsulation methods (KEMs) have been defined for this purpose, and they generally offer better (tighter) security proofs than the more general-purpose PK techniques above, see Chapter 16. There are also data encapsulation methods (DEMs) which provide an "envelope" for more arbitrary data. When implementing hybrid encryption, the considerations discussed in Section 8.4.4 apply.

## 13.2 RSA/Factoring Based

### 13.2.1 RSA PKCS#1 v1.5

**Definition:** RFC 3447, [95] (see also [164]).

**Parameters:** integer $N$, product of primes $p, q$, encryption exponent $e$, decryption exponent $d$. $N, e$ are public.

**Security:** Relies on the intractability of the RSA problem (and thus integer factoring), but security *may* be less.

**Deployment:** Widespread (e.g. (W)TLS, S/MIME).

**Implementation:**

**Public analysis:** Cryptrec [46], papers such as those listed in "known weakness" below.

**Known weakness:** Bad choices of $p, q$ exist but are (w.h.p.) avoided by random choices for $p, q$ of roughly the same size. Small $e$ may open up to attacks on related messages, [42, 43, 78] (e.g. similar messages, re-using same random pads). Implementations that allow extensive adaptive chosen ciphertext attacks (about one million messages for a 1024-bit key), reporting back decryption errors, can be exploited to recover messages, [24].

**Comments:** Due to lack of security proof and the known vulnerabilities, we recommend whenever possible to use RSA-OAEP, or preferably, hybrid encryption based on RSA-KEM. If used, we recommend $|N| \geq 1024$ for legacy systems and $|N| \geq 2432$ for new systems. We recommend, if possible, to use large, random $e$, and/or to restrict use to protection of short, random messages. We recommend not to use the same keys for encryption and signatures. Multi-prime options, using more than two primes exists, are defined in [164] and key-sizes/number of factors are in this case analyzed in [114].

### 13.2.2 RSA-OAEP

**Definition:** ISO/IEC 18033-2, [88] (see also [164, 95]).

**Parameters:** as for PKCS#1v1.5, plus a hash function and a mask generating function (MGF).

**Security:** provably as secure as RSA inversion in the ROM, but with a loose reduction, [59].

**Deployment:** unclear, though included in several other standards, such as IEEE Std 1363, S/MIME, etc.

**Implementation:**

**Public analysis:** NESSIE [156], Cryptrec [46], security proof [59] (correction of [17]).

**Known weakness:** Bad choices of $p, q$ exists but are (w.h.p.) avoided by random choices for $p, q$ of roughly the same size. Implementations must not enable attackers to distinguish between error conditions arising during the decoding process, since it may open up efficient chosen ciphertext attacks, [123].

**Comments:** If used, we recommend $|N| \geq 1024$ for legacy systems and $|N| \geq 2432$ for new systems. In addition we recommend to use the SHA-family of hash functions rather than MD2, MD5 (part of the PKCS#1 specification). Pending investigation of the exact impacts of recent progress in cryptanalysis of SHA-1 (see Section 10.6), somewhat cautious use of SHA-1 may also be advisable. Due to the tighter security proof and the simpler implementation, we recommend to consider RSA-KEM together with hybrid encryption as an alternative to OAEP, see Chapter 16.

#### Discussion

This is the new RSA PKCS#1v2.0, included also in e.g. ISO/IEC 18033-2. Security is related, in the random oracle model, to the "RSA problem" with a very loose reduction, [59]. This is because of the quadratic-time cost of the reduction from the partial-domain RSA problem and a success probability lost from the RSA problem.

From an efficiency point of view, encryption and decryption are very similar to the plain RSA cryptosystem, except two more hashings. The advantage with respect to RSA-KEM (see Section 16.2.1) is the size of the produced ciphertexts, which, in this case, is just an element in $\mathbb{Z}_N^*$.

Suggestion to consider for future standards development: OAEP 3-round. Instead of a 2-round Feistel network, one can make a 3-round construction. This improves the reduction, since RSA-OAEP-3R directly reduce from the "RSA problem" (still with a quadratic cost). Furthermore, bandwidth can be improved since redundancy (the zeroes in OAEP) is no longer required.

## 13.3 ElGamal/Discrete log based

We do not present any general purpose DLOG based schemes for short messages. However, the document does present a so-called hybrid scheme, namely ECIES. This is obtained by

applying the ECIES KEM from Chapter 16 and then applying a suitable form of symmetric key based authenticated encryption (i.e. a DEM).

# Chapter 14

# Signatures

## 14.1 Overview

A signature scheme is usually defined as a triplet of algorithms $(K, S, V)$, where $K$ is the Key generation algorithm, $S$ is the Signing algorithm and $V$ is the Verification algorithm. K generates pairs $(s, v)$ of keys for the Signing/Verification algorithm. Only the party knowing $s$ is able to generate a valid signature on $m$, $\sigma(m)$, but using $V$ and the corresponding key $v$ (assumed to be public information), anybody can efficiently decide if a given $(m, \sigma(m))$ pair is valid. Note that some schemes, e.g. Identity based schemes, also specify a fourth function, $P$, that generates parameters.

To meet security requirements (more on this below) and to allow signing of more or less arbitrary long messages, a signature scheme requires a hash function, so that the signing/verification algorithms operate on a fixed-size hash of the message. The combination of signature algorithm and hash function is called a *signature suite*. As discussed earlier, the hash output should be twice the "security level" (in bits). In principle, any secure hash/signature combination could be used. However, some issues should be brought to attention. For DSS, it does not make sense to use e.g. the MD5 hash algorithm, since DSS by definition only allows the SHA-family. Even if no security issue can be seen, it would create a completely new "DSS", which increases complexity and reduces interoperability. Secondly, a too liberal use of different hash functions may open up so-called bidding-down attacks, where the security corresponds to the *weakest* available hash function. In particular, it is important to tie the signature value to the hash function used in creating it, see [99].

Some signature schemes enable the whole message, or part of it, to be recovered from the signature. These schemes can be useful in constrained environments because only the non-recoverable part of the message need be stored or transmitted with the signature.

As for asymmetric encryption, main choices are whether to use factoring or DLOG based schemes (in the latter case also which group) and what security model/proof (if any) one finds attractive.

### 14.1.1 Security Notions

The today most widely used security notion for signatures is similar to that for MACs, and is called *resistance against existential forgery under adaptive chosen message attack*. Informally, this means that the attacker is allowed to have messages of his own choosing signed by a "signing oracle", after which the attacker is to provide a single valid $(m, \sigma(m))$-pair that he

has not seen before. Below, unless otherwise noted, "secure" is used in this sense. Again, for a more extensive discussion of the security of asymmetric schemes, we refer to [156].

## 14.2 RSA/Factoring Based

### 14.2.1 RSA PKCS#1 v1.5

**Definition:** RFC 3447, [95] (see also [164]).

**Parameters:** integer $N$, product of primes $p, q$, a signing exponent $d$, verification exponent $e$ where $N, e$ are public, and a hash function.

**Security:** Relies on the intractability of the RSA problem (and thus integer factoring), but security *may* be less.

**Deployment:** Widespread (e.g. (W)TLS, S/MIME).

**Implementation:**

**Public analysis:** Cryptrec [46].

**Known weakness:** Bad choices of $p, q$ exists but are (w.h.p.) avoided by random choices for $p, q$ of roughly the same size. $d$ must not be small.

**Comments:** We recommend to use at least 160-bit hash functions and $|N| \geq 1024$ for legacy systems, or for new deployments we recommend 224-bit hashes and $|N| \geq 2432$. A public exponent of $e > 65536$ is recommended, smaller $e$ may be used if performance is critical. However, due to the lack of security proof, we recommend whenever possible to use RSA PSS instead, there is no advantage in using v1.5. We recommend not to use the same keys for encryption and signatures, nor using the same key with both RSA PSS and v1.5.

As already discussed in Chapter 10, we recommend to phase-out the use of the MD5 hash function and SHA-1 should be avoided in new deployments.

### 14.2.2 RSA-PSS

**Definition:** RFC 3447, [95] (see also [164]).

**Parameters:** as for v1.5 plus a mask generating function (MGF).

**Security:** Provably as secure as the RSA problem in the ROM (tight reduction) [94].

**Deployment:** Proposed for inclusion in standards (e.g. IEEE Std 1363), but deployment/use is unclear.

**Implementation:**

**Public analysis:** NESSIE and Cryptrec reports [156, 46], several papers, e.g. [94, 44].

**Known weakness:** same as for v1.5.

**Comments:** Recommended alternative to v1.5, but we recommend not to use the same key. We recommend to use at least 160-bit hash functions and $|N| \geq 1024$ for legacy systems, or for new deployments we recommend 224-bit hashes and $|N| \geq 2432$. A public exponent of $e > 65536$ is recommended, though small $e$ may be used if performance is critical. We recommend against the use of the MD5 hash function. RSA PSS-R, giving message recovery, is specified in ISO/IEC 9796-2.

As already discussed in Chapter 10, we recommend to phase-out the use of the MD5 hash function and SHA-1 should be avoided in new deployments.

## 14.3 ElGamal/Discrete Log Based

### 14.3.1 DSA

**Definition:** FIPS PUB 186-2 (part of DSS), [142].

**Parameters:** $p$, an $L = 1024$ bit prime defining a prime field, and $q$, a 160-bit prime divisor of $p - 1$ defining a cyclic subgroup. Parameters can be generated according to method described in standard the [142]. For future deployments we recommend a $p$ with at least $L = 2432$ with has a $l = 224$-bit prime $q$ dividing $p - 1$.

**Security:** for proper choice of parameters, the best known attack has roughly the same complexity as that of factoring $L$-bit numbers, or, $2^{l/2}$, whichever is smaller.

**Deployment:** Widespread. (Included in numerous standards, e.g. IKE, (W)TLS, IEEE Std 1363, ISO/IEC 9796-3, etc.)

**Implementation:** Test vectors available in [142].

**Public analysis:** several papers, e.g. [153, 194, 195] (details below), Cryptrec [46].

**Known weakness:** sensitive to "misuse", e.g. predictable nonces [153], malicious parameter generation [194], but easily avoided. An unpublished attack exploiting "skewness" of the distribution of the random number generator from FIPS 186 claims a workfactor of $2^{64}$ and requires $2^{22}$ known signatures when $(L, l) = (1024, 160)$. A remedy is suggested in FIPS 186-2.

**Comments:** FIPS 186-2 defines the Digital Signature Standards (DSS) which specifies both the signature algorithms as well as the use of the SHA-1 hash function. For interoperability reasons, we do not recommend the use of DSA with other hash functions. FIPS 186-3 will specify larger keys and hashes. There exists variants on DSA, e.g. G(erman)DSA, K(orean)DSA, etc. While we see no security problems with these, we recommend the use of DSA to limit the number of options.

We recommend to verify the correctness of parameters to mitigate attacks such as those mentioned above. Even if the parameters have been already been "certified", this gives some extra protection. However, it has been shown that the method to generate a random curve (or rather the one to prove that a curve was generated at random) is flawed [195].

As already discussed in Chapter 10, SHA-1 should be avoided in new deployments.

### 14.3.2 ECDSA

**Definition:** ANSI X9.62, [8] (part of DSS [142] and Suite-B [157], also in SECG, IKE, (W)TLS, IEEE Std 1363, ISO 14888-3, 15946-2, etc.).

**Parameters:** a subgroup over an elliptic curve defined over a prime or binary field. The standard [142] defines 4 curves over prime fields and 8 over binary fields, the curves corresponding roughly to $L = 160, 256, 384$, and 512-bit keys. A method to generate random curves is also specified (taken from [8]).

**Security:** for proper choice of parameters and $L$-bit keys, best known attack has complexity about $2^{L/2}$. In the generic group model, better attacks can be excluded, see [32]. However, the relevance of the generic group model for ECDSA has been questioned, [189] (see also "duplicate signatures" below).

**Deployment:** Widespread.

**Implementation:** Test vectors and implementation hints available in [142].

**Public analysis:** As for DSA, plus the NESSIE and Cryptrec reports [156, 46] the papers [32, 154].

**Known weakness:** As for DSA, in particular the possibility of badly generated curves over binary fields [195]. We recommend to avoid bad parameters choices by either using the specified curves in [142], or, to use random curves according to [8].

**Comments:** As for DSA. Note that also SHA-256, 384 and 512 may be required to give matching security. The use of Koblitz curves over binary fields requires slightly larger keys (the best attacks are for curves over the field $\mathbb{F}_{2^m}$ about $\sqrt{2m}$ times faster, usually not critical). Some general concerns exist about possible future attacks on curves of "special form", or over "special fields" e.g. [205]. As a first choice, we recommend curves over prime fields. Existence of duplicate signatures: note that each message has two valid signatures, if $(r, s)$ is a valid signature, then so is $(r, -s)$, which in a sense means that DSA EC groups cannot be considered "generic". Finally, an "interoperability chain" for various ECDSA standards is noted in [93]: FIPS 186-2 < ANSI X9.62 < IEEE Std 1363-2000 < ISO 14888-3, i.e. an implementation according to FIPS 186-2 is also compliant with ANSI X9.62, etc, but the converse may not hold. (Note however that X9.62 and 14888-3 have been revised since [93] was written and it is therefore to be confirmed if this still holds.)

For legacy systems we recommend using a curve with a prime order subgroup with at least 160 bits, however for new deployments we recommend choosing groups with prime order of size at least 224-bits.

We recommend to verify the correctness of parameters to mitigate attacks such as those mentioned above. Even if the parameters have been already been "certified", this gives some extra protection.

As already discussed in Chapter 10, SHA-1 should be avoided in new deployments.

# Chapter 15

# Public Key Authentication and Identification

## 15.1 Overview

In this primitive category we find protocols between a *prover* and a *verifier*. The purpose of these protocols is for the prover to be able to convince the verifier that "he is who he claims to be". First, the protocol should be *complete*: if the prover indeed is who he claims to be, he should be able to convince any verifier (who wishes to be convinced) that this is the case. Secondly, the protocol should be *sound*: nobody but the real prover should be able convince anybody of this fact. Put differently, "convincing" here means ability to prove knowledge of some secret information, that is in some correspondence to some publicly available information such as an identity/public key, available to the verifier.

At present, the only included scheme is the GQ-scheme by Guillou and Quisquater, which is a *zero knowledge identification protocol*. Intuitively, this means that no other information, except the fact that the other party is who he/she claims to be, is revealed. In particular, no information about the secret key is leaked.

Note that with any identification protocol, the identity/authenticity of the other party is only assured at the time of protocol execution. Thus, if granting access or service is based on success of the protocol, a trusted path must also be in place, or be cryptographically established by tying together the identification with key agreement and integrity protection.

Again, for a more extensive discussion of the security of asymmetric identification schemes, we refer to [156].

## 15.2 GQ

**Definition:** ISO/IEC 9798-5, [85].

**Parameters:**

**system-wide:**

- public: an RSA public key, $(N, e)$, where $e$ is prime
- secret (known only to "system authority"): a secret RSA exponent, $d$ (corresponding to $e$)

**user-specific:**

- public key: an integer $G$
- secret key: integer $Q$ (such that $G \equiv Q^{-e} \pmod{N}$)

**security parameter:** $t$, the number of repetitions

**Security:** asymptotically, if $te < A \log^B n$ (for some constants $A, B$), the scheme is (perfect) zero knowledge and, if a certain (strengthened) RSA assumption holds, the forgery success of an attacker is bounded by $e^{-t}$, see [33, 66, 67, 16].

**Deployment:** unclear

**Implementation:**

**Public analysis:** NESSIE, [156].

**Known weakness:** as with other known ZK identification protocols, implementations in environments where it is possible to "reset" the prover (e.g. removal of power from smart-card) may fail to provide security.

**Comments:** The scheme can be made identity based by letting $G = f(ID)$, where $ID$ encodes some user-identity, and $f$ is a suitable function. A usual recommendation is to use $t = 1$ so that the choice of $e$ directly determines the security level. Since factorization of $N$ is unknown, attacks based on fault analysis of CRT implementations in end-user devices can be excluded. Also, some side-channel attacks can be excluded due to the use of public exponents, [156].

# Chapter 16

# Key Encapsulation Mechanisms

## 16.1 Overview

The modern method for using public key encryption is by so-called hybrid schemes. These combine a public key method for transmitting a symmetric key (a key encapsulation mechanism or KEM) with a method for encrypting the payload via a symmetric key based mechanism (a data encapsulation mechanism or DEM). The KEM-DEM methodology allows for highly efficient schemes, and a modular approach to new scheme development. The standard ISO/IEC 18033-2 [88] specifies complete algorithm suites for hybrid encryption.

The security notion used for the asymmetric primitive in a KEM is basically the same as for general public key encryption, though considering that the protected value is to be used as a key, a secure key derivation function is also needed. Similarly, for Diffie-Hellman key-agreement variants, security means that observing the exchanged public values should reveal no useful information about the agreed secret.

A central question in cryptography is the relation between the Diffie-Hellman problem (CDH), its variants (e.g. DDH), and the DLOG problem. Some partial results, stating assumptions/conditions under which equivalence holds/does not hold, can be found in e.g. [126, 127, 128].

As with most public key schemes, various attacks based on timing analysis, power analysis, or fault analysis may need to be considered, depending on the deployment environment, see e.g. [156].

## 16.2 Factoring Based

### 16.2.1 RSA-KEM

**Definition:** ISO/IEC 18033-2, [88].

**Parameters:** same as for the basic RSA scheme $(N, e, d)$, plus a secure key derivation function (KDF).

**Security:** provably as secure as RSA inversion, very tight reduction with KDF modeled in the ROM, see e.g. [182].

**Deployment:** unclear

**Implementation:**

**Public analysis:** NESSIE [156]. The paper [79] shows that (asymptotically) any block of $O(\log \log N)$ bits is as secure as whole message (loose reduction).

**Known weakness:** same caveats as for general RSA key generation, easily avoided.

**Comments:** We and $|N| \geq 1024$ for legacy systems, or for new deployments we recommend $|N| \geq 2432$. A public exponent of $e > 65536$ is recommended, though smaller $e$ may be used if performance is critical. Due to the homomorphic properties of RSA, security requirements on the KDF may be higher than some other schemes. We recommend following the ISO 18033-2 specification for KDFs. RSA-KEM is an improvement of RSA-REACT. Encryption can be quite efficient, while decryption requires an inversion of the RSA function. As consequence the scheme remains very similar to the plain RSA cryptosystem (from an efficiency point of view).

## 16.3 DLOG Based

### 16.3.1 ECIES-KEM

**Definition:** ISO/IEC 18033-2, [88] (Also see ANSI X9.63 [9] and SECG [172])

**Parameters:** a cyclic elliptic curve group of size $q$, and a KDF.

**Security:** modeling the KDF in the ROM, the scheme is provably as secure as the gap-DDH problem. If one models the group as a generic group then it is secure on the basis of the security of the hash function. See [48] for a full discussion.

**Deployment:** unclear

**Implementation:**

**Public analysis:**

**Known weakness:** There are some legacy issues with earlier standards with respect to choices of the associated DEM and with issues related to "benign" malleability. The first of these is easily overcome, whilst the second is a matter of ongoing discussion as to whether it is a security weakness or a feature.

**Comments:** Normal security considerations for parameter choice applies. We recommend to use a suitable elliptic curve with a group order divisible by a prime $q$ of at least 160-bits for legacy systems, or 224-bits for new deployments.

### 16.3.2 PSEC-KEM

**Definition:** ISO/IEC 18033-2, [88].

**Parameters:** a cyclic group (subgroup of a finite field or elliptic curve) of size $q$, and a KDF.

**Security:** modeling the KDF in the ROM, the scheme is provably as secure as the CDH problem with a tight reduction, see [182].

**Deployment:** unclear

**Implementation:**

**Public analysis:** NESSIE [156].

**Known weakness:**

**Comments:** Normal security considerations for parameter choice applies. We recommend to use prime fields of at least 1024 (resp. 2432) bits or a suitable elliptic curve, in both cases with at least 160 (resp. 224) bit $q$ for legacy (resp. new) systems. Binary fields may be used if performance is an issue.

### 16.3.3 ACE-KEM

**Definition:** ISO/IEC 18033-2, [88].

**Parameters:** as for PSEC-KEM, plus a hash function

**Security:** under the assumption that the hash is 2nd preimage resistant and that the KDF has certain pseudorandom properties, the scheme is provably as secure as the DDH problem with a tight reduction (i.e. in the standard model), see [45].

**Deployment:** unclear

**Implementation:**

**Public analysis:** NESSIE [156].

**Known weakness:**

**Comments:** Normal security considerations for parameter choice applies. We recommend to use prime fields of size at least 1024 (resp. 2432) bits or a suitable elliptic curve, in both cases with at least 160 (resp. 224) bit $q$ and hash function for legacy (resp. new) systems. Binary fields may be used if performance is an issue. Different security proofs under different assumptions are known, see [156] for an overview. In particular it has been shown in [45] that ACE-KEM is at least as secure as ECIES-KEM (which is also included in ISO 18033-2). Implementations should not reveal cause of decryption errors. Doing so does not obviously open up for attacks, but the security proof no longer holds.

# Chapter 17

# Key Agreement and Key Distribution

## 17.1 Overview

Many protocols make use of key agreement/transport sub-protocols. These come in essentially three variants. Either they are symmetric key based (such as Kerberos), or they engage in a simple public key key-transport mechanism (akin to RSA-KEM from Chapter 16 and used in early deployed versions of SSL), or they use a forward-secure key agreement scheme based (usually) on the Diffie–Hellman protocol. The use of Diffie–Hellman based key agreement in SSL/TLS implementations is becoming more widespread due to security and legal concerns, it is often the default setting for modern browser/web-server combinations.

Some protocols/applications allow for DH to be used without authentication, often referred to as *opportunistic mode*. Note that without authentication, DH is totally vulnerable to an active man-in-the-middle attack: Alice and Bob think they have exchanged a key, but in reality there has been created one key between Alice and the attacker, and one key between the attacker and Bob. Consequently, the attacker can eavesdrop on all communication, acting as relay between Alice and Bob. Therefore, DH may fail to provide any security when active attacks are possible. Thus DH is often used in combination with some form of authentication mechanism, often these are digital signatures (in the case of SSL/TLS) althought this needs to be done carefully due to possible unknown-key-share attacks. Variants of DH exists (such as MQV, HMQV, MTI etc) which combine the authentication into the key derivation, these variants provide implicit authentication and are often more efficient than a method which combines DH with a standard signature algorithm.

While DH is often a quite heavy-weight protocol, it has one advantage: *perfect forward secrecy*. This means that breaking one DH-created key still does not help an attacker breaking other DH-created keys.

#### DH According to the Internet Key Exchange (IKE)

**Definition:** RFC 2409, [72].

**Parameters:** The IKE protocol suggests four cyclic groups (subgroup of a finite field or elliptic curve) of different sizes for key exchange. Later, 6 more prime field groups have

been added [107]. The choice of the original groups originates from the Oakley protocol [160]. A secure PRF is also required.

**Security:** as claimed.

**Deployment:** Widely used to secure Internet connections.

**Implementation:** NIST PlutoPlus reference implementation for Linux [151].

**Public analysis:** The security of Oakley groups 3 and 4 are analyzed in [185] without finding any practical attack or weaknesses, although their use is not recommended.

**Known weakness:** The Oakley groups 1, 2 and 3 are too small to provide adequate security.

**Comments:**

# Bibliography


[1] 3GPP TS 35.202, *Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi algorithm specification*, available from `http://www.3gpp.org/ftp/Specs/html-info/35202.htm`.

[2] 3GPP TS 35.203, *Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 3: Implementors' Test Data*, available from `http://www.3gpp.org/ftp/Specs/html-info/35202.htm`.

[3] 3GPP TS 35.204, *Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 4: Design Conformance Test Data*, available from `http://www.3gpp.org/ftp/Specs/html-info/35204.htm`.

[4] 3GPP TR 33.908, *3G Security; General report on the design, specification and evaluation of 3GPP standard confidentiality and integrity algorithms*, available from `http://www.3gpp.org/ftp/Specs/html-info/33908.htm`.

[5] F. Almgren, G. Andersson, T. Granlund, L. Ivansson, and S. Ulfberg, *How we Cracked the Code Book Ciphers,* Report, 2000. Available via answers.codebook.org

[6] E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir and S. Zimmer, *Second Preimage Attacks on Dithered Hash Functions*, Proceedings of Eurocrypt 2008, LNCS **4965**, pp. 270–288, Springer-Verlag.

[7] ANSI X9.19-1996, *Financial Institution Retail Message Authentication.*

[8] ANSI X9.62, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).*

[9] ANSI X9.63, *Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography.*

[10] The AES Lounge, `http://www.iaik.tu-graz.ac.at/research/krypto/AES/index.php`.

[11] K. Aoki, Y. Kida, T. Shimoyama, H. Ueda, *Subject: SNFS274*, Announcement, 24 Jan 2006.

[12] F. Bahr, M. Boehm, J. Franke, T. Kleinjung, *Subject: RSA200*, Announcement, 9 May 2005.

[13] M. Bellare, *New Proofs for NMAC and HMAC: Security Without Collision-Resistance*, Proceedings of Crypto 06, LNCS **4117**, pp. 602–619, Springer-Verlag, 2006.

[14] M. Bellare, R. Canetti, and H. Krawczyk, *Keying hash functions for message authentication*, Proceedings Crypto 96, LNCS **1109**, Springer-Verlag, 1996. Full paper available at `http://www.cs.ucsd.edu/users/mihir/papers/hmac.html`.

[15] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, *A Concrete Treatment of Symmetric Encryption: Analysis of DES Modes of Operation,* Proceedings of 38th IEEE FOCS, pp. 394–403, 1997.

[16] M. Bellare and A. Palacio, *GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks*, Proceedings of Crypto'02, LNCS **2442**, pp. 162–177, Springer-Verlag.

[17] M. Bellare and P. Rogaway, *Optimal asymmetric encryption (how to encrypt with RSA)*, Proceedings of Eurocrypt'94, LNCS **950**, pp. 92–111, Springer-Verlag.

[18] S. Bellovin, *Problem Areas for the IP Security Protocols*, Proceedings of the 6th Usenix Unix Security Symposium, pp. 1–16, 1996, available at www.research.att.com/~smb/papers/index.html

[19] D. Bernstein, *Circuits for Integer Factorization: A Proposal,* Manuscript, Nov. 2001. Available via `http://cr.yp.to/papers.html`.

[20] E. Biham, O. Dunkelman, and N. Keller, *A Related-Key Rectangle Attack on the Full KASUMI*, Proceedings of ASIACRYPT 2005, LNCS **3788**, pp. 443–461, 2005.

[21] E. Biham and A. Shamir, *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, **4** (1991), 3–72.

[22] E. Biham and Y. Carmeli, *Efficient Reconstruction of RC4 Keys from Internal States*, Proceedings of FSE 2008, LNCS **5086**, 2008.

[23] J. Black, *Authenticated encryption*, In "Encyclopedia of Cryptography and Security", Springer-Verlag, 2005.

[24] D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1*, Proceedings of Crypto'98, LNCS **1462**, pp. 1–12, Springer-Verlag.

[25] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, *Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security,* Report of ad hoc panel of cryptographers and computer scientists, Jan. 1996. Available via `http://www.crypto.com/papers/`.

[26] The block cipher lounge, `http://www2.mat.dtu.dk/people/Lars.R.Knudsen/bc.html`.

[27] The Blowfish page, `http://www.schneier.com/blowfish.html`.

[28] M. Blunden and A. Escott, *Related Key Attacks on Reduced Round KASUMI*, Proceedings of FSE 2001, LNCS **2355**, pp. 277–285, Springer-Verlag.

[29] M. Bodén and S. Kowalski, *Value based risk analysis: the key to successful commercial security targets for the Telecom Industry*, Proceedings of 2nd Common Criteria Conference, 2002.

[30] G. Brassard, P. Hoyer, A. Tapp, *Quantum cryptanalysis of hash and claw-free functions*, ACM SIGACT, **28**:2, 1997, 14–19.

[31] J. R. T. Brazier, *Possible NSA Decryption Capabilities,* Manuscript 1999, available via jya.com/nsa-study.htm

[32] D. R. L. Brown, *Generic Groups, Collision Resistance, and ECDSA*, available at `http://eprint.iacr.org/2002/026/`.

[33] M. Burmester, *An almost-constant round interactive zero-knowledge proof*, Information Processing Letters, **42**:2, 81–87, 1992.

[34] R. Canetti, O. Goldreich, and S. Halevi, *The Random Oracle Methodology, Revisited*, In Proceedings of 30th Annual ACM Symposium on the Theory of Computing, pp. 209–218, May 1998, ACM.

[35] C. De Canniére and C. Rechberger, *Finding SHA-1 Characteristics: General Results and Applications*, Proceedings of ASIACRYPT 2006, LNCS **4284**, pp. 1–20, Springer-Verlag.

[36] C. De Canniére and C. Rechberger, *SHA-1 collisions: Partial meaningful at no extra cost?*, Presented at rump session of CRYPTO 2006.

[37] C. De Cannière and C. Rechberger, *Finding SHA-1 Characteristics*, NIST - Second Cryptographic Hash Workshop, 2006.

[38] C. De Cannière, F. Mendel and C. Rechberger, *Collisions for 70-Step SHA-1: On the Full Cost of Collision Search*, Proceedings of SAC 2007, LNCS **4876**, pp. 56–73, Springer-Verlag.

[39] C. De Cannière and C. Rechberger, *Preimages for Reduced SHA-0 and SHA-1*, Proceedings of CRYPTO 2008, LNCS **5157**, pp. 179–202, Srpinger-Verlag.

[40] I. L. Chuang, N. Gershenfeld, and M. Kubinec, *Experimental Implementation of Fast Quantum Searching,* Physical Review Letters, **80**:15 (1998), 3408–3411.

[41] S. Contini and Y. L. Yin, *Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions*, Proceedings of ASIACRYPT 2006, LNCS **4284**, pp. 37–53, Springer-Verlag.

[42] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *Low-Exponent RSA with Related Messages*, Proceedings of Eurocrypt '96, LNCS **1070**, pp. 1–9, Springer-Verlag.

[43] J.-S. Coron, M. Joye, D. Naccache and P. Paillier, *New Attacks on PKCS #1 v1.5 Encryption*, Proceedings of Eurocrypt 2000, LNCS **1807**, pp. 369–379, Springer-Verlag.

[44] J.-S. Coron, M. Joye, D. Naccache and P. Paillier, *Universal Padding Schemes for RSA*, Proceedings of Crypto 02, LNCS **2442**, pp. 226–241, Springer-Verlag.

[45] R. Cramer and V. Shoup, *Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack*, Cryptology ePrint Archive, Report 2001/108, 2001.

[46] Cryptrec report annual 2002, available at `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/c02e\_report2.pdf`.

[47] M. Daum and S. Lucks, *Attacking Hash Functions by Poisoned Messages, "The Story of Alice and her Boss"*, available at `http://http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/`.

[48] A.W. Dent. *Proofs of security for ECIES*. Chapter III of *Advances in Elliptic Curve Cryptography*, pp 41–46, Cambridge University Press, 2005.

[49] I. Devlin and A. Purvis, *A fundamental evaluation of 80 bit keys employed by hardware oriented stream ciphers*, Workshop record of SHARCS 2006, www.ruhr-uni-bochum.de/itsc/tanja/SHARCS/

[50] I. Devlin and A. Purvis, *Assessing the Security of Key Length*, Workshop record of SASC 2007, sasc.crypto.rub.de/program.html

[51] distributed.net, *Project RC5,*
available via `http://www.distributed.net/rc5/`.

[52] ECRYPT NoE, *Recent Collision Attacks on Hash Functions: ECRYPT Position Paper*, ECRYPT document STVL-ERICS-2-HASH_STMT-1.1, Feb. 2005, available at `http://www.ecrypt.eu.org/documents/STVL-ERICS-2-HASH\_STMT-1.1.pdf`.

[53] ECRYPT NoE, *AES Security Report*, ECRYPT deliverable D.STVL.2, Jan 2006, available at `http://www.ecrypt.eu.org/documents/D.STVL.2-1.0.pdf`.

[54] ECRYPT NoE, *eHash home page*, `http://ehash.iaik.tugraz.at`.

[55] ECRYPT NoE, *eStream home page*, `http://www.ecrypt.eu.org/stream`.

[56] EFF, Website of the electronic frontier foundation. `http://www.eff.org/descracker.html`.

[57] ETSI TS 102 176, *Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures*, ETSI, Nov 2004.

[58] P.-A. Fouque, G. Leurent, P. Q. Nguyen, *Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5*, Proceedings of Crypto 2007, **4622**, pp. 13–30, Springer-Verlag.

[59] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, *RSA-OAEP is secure under the RSA assumption*, Proceedings of Crypto'01, LNCS **2139**, pp. 260–274, Springer-Verlag.

[60] C. Gaj et al., *Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Arrays*, In *CT-RSA 2001*, LNCS **2020**, pp. 84—99.

[61] C. Gehrmann and K. Nyberg, *Security in Personal Area Networks*, In C. Mitchell (Ed.): Security for Mobility, IEE 2003.

[62] H. Gilbert and H. Handschuh. *Security analysis of SHA-256 and sisters*, Proceedings of SAC 2003, LNCS **3006**, pp. 175–193, Springer-Verlag.

[63] J. Gilmore (Ed.), *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design,* Electronic Frontier Foundation, O'Reilly & Associates, 1998.

[64] L. Granboulan, *How to repair ESIGN*, Proceedings of SCN '02, also available at http://eprint.iacr.org/2002/074, 2002.

[65] L. K. Grover, *A fast quantum mechanical algorithm for database search,* Proceedings of the 28th ACM STOC, pp. 212–219, 1996.

[66] L. C. Guillou and J.-J. Quisquater, *A "paradoxical" identity-based signature scheme resulting from zero-knowledge*, Proceedings of Crypto'88, LNCS **403**, pp. 216–231, Springer-Verlag, 1988.

[67] L. C. Guillou and J.-J. Quisquater, *A practical Zero-Knowledge protocol fitted to security microprocessor minimizing both transmission and memory*, Proceedings of Eurocrypt'88, LNCS **330**, pp. 123–128, Springer-Verlag.

[68] S. Halevi and H. Krawczyk. *Strengthening Digital Signatures via Randomized Hashing.* Proceedings of Crypto 2006, LNCS **4117**, pp. 41–59, Springer-Verlag.

[69] I. Hamer and P. Cho, *DES Cracking on the Transmogrifier 2a*, In Ç. K. Koç and C. Paar (Eds.), *Cryptographic Hardware and Embedded Systems, 1st International Workshop, CHES 1999 Proceedings*, LNCS **1717**, pp. 13–24. Springer-Verlag.

[70] H. Handschuh and B. Preneel, *Minding your MAC algorithms*, Draft 2004, Submitted to ISB Journal.

[71] H. Handschuh and B. Preneel, *Key-Recovery Attacks on Universal Hash Function based MAC Algorithms*, Proceedings of Crypto 2008, LNCS **5157**, pp. 144–161, Springer-Verlag.

[72] D. Harkins and D. Carrel, *The Internet Key Exchange (IKE)*, RFC 2409, IETF.

[73] P. Hawkes, M. Paddon, and G. Rose, *On corrective patterns for the SHA-2 family*, Cryptology ePrint Archive, Report 2004/207, August 2004. http:// eprint.iacr.org/

[74] Hash function lounge, paginas.terra.com.br/informatica/paulobarreto/hflounge.html

[75] Helion, Website: `http://www.heliontech.com/`.

[76] A. Hodjat and I. Verbauwhede, *A 21.54 gbits/s fully pipelined AES processor on FPGA*, In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines.*

[77] J. Hoffstein, J. Pipher, and J. H. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem,* In *Proceedings of ANTS III*, LNCS **1423**, pp. 267–288, Springer-Verlag, 1998.

[78] J. Håstad, *Solving Simultaneous Modular Equations of Low Degree*, SIAM J. of Computing, **17**, 336–341, 1988.

[79] J. Håstad and M. Näslund, *The security of all RSA and discrete log bits*, J. ACM **51**:2, 187–230 (2004).

[80] http://www.weizmann.ac.il/~itsik/RC4/rc4.html

[81] http://burtle.burtle.net/bob/rand/isaac.html

[82] http://planeta.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html

[83] S. Indesteege, F. Mendel, B. Preneel and C. Rechberger, *Collisions and other Non-Random Properties for Step-Reduced SHA-256*, Proceedings of SAC 2008 (to appear).

[84] ISO/IEC 9797-1:1999, *Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher.*

[85] ISO/IEC 9798-5:2004, *Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero knowledge techniques.*

[86] ISO/IEC 10116:2006, *Information technology — Security techniques — Modes of operation for an n-bit block cipher.*

[87] ISO/IEC 10118-3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions.*

[88] ISO/IEC 18033-2:2006, *Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric Ciphers.*

[89] ISO/IEC 18033-3:2005, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers.*

[90] ISO/IEC 18033-4:2005, *Information technology — Security techniques — Encryption algorithms — Part 4: Stream ciphers.*

[91] IEEE Std 1363-2000, *Standard Specification for Public-Key Cryptography.*

[92] T. Iwata. On the Impact of Key Check Value on CBC MACs. Seminar 09031 on "Symmetric Cryptography", Schloss Dagstuhl, January 2009.

[93] D. Johnson, A. Menezes, and S. Vanstone, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Submission to NESSIE.

[94] J. Jonsson, *Security proofs for the RSA-PSS signature schemes and its variants*, available at http://eprint.iacr.org/2001/053/, 2001.

[95] J. Jonsson, B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, RFC 3447, IETF.

[96] A. Joux, *Multicollisions in iterated hash functions, application to cascaded constructions,* Proceedings of Crypto 04, LNCS **3152**, pp. 306–316, Springer-Verlag, 2004.

[97] A. Joux and T. Peyrin, *Hash functions and the (amplified) boomerang attack*, ECRYPT Hash Workshop, Barcelona, Spain, 2007.

[98] A. Joux, D. Naccache, E. Thoé, *When e-th Roots Become Easier Than Factoring*, Proceedings of Asiacrypt 2007, LNCS **4883**, pp. 13–28, Springer-Verlag.

[99] B. Kaliski, *Hash Function Firewalls in Signature Schemes*, RSA Conference 2002, LNCS **2271**, pp. 1–16, Springer-Verlag.

[100] B. Kaliski, *TWIRL and RSA Key Size,* Available via www.rsasecurity.com/rsalabs

[101] J. S. Kang, S. U. Shin, D. Hong, and O. Yi, *Provable security of KASUMI and 3GPP encryption mode f8*, Proceedings of ASIACRYPT 2001, LNCS **2248**, pp. 255–271, Springer-Verlag.

[102] O. Kara and C. Manap, *A New Class of Weak Keys for Blowfish*, Proceedings of FSE 2007, LNCS **4593**, pp. 167–180, Springer-Verlag.

[103] J. Kelsey and B. Schneier, *Second Preimages on n-Bit Hash Functions for Much Less than $2^n$ Work*, Proceedings of EUROCRYPT 2005, LNCS **3494**, pp. 474–490, Springer-Verlag.

[104] J. Kelsey and T. Kohno, *Herding Hash Functions and the Nostradamus Attack*, Proceedings of EUROCRYPT 2006, LNCS **4004**, pp. 183–200, Springer-Verlag.

[105] T. Kerins, E. Popovici, A. Daly and W. Marnane, *Hardware encryption engines for e-commerce*, In *Proceedings of Irish Signals and Systems Conference, ISSC 2002*, pp. 89–94.

[106] J. Kim, A. Biryukov, B. Preneel, and S. Hong, *On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1*, Proceedings of SCN, LNCS **4116**, pp. 242–256, Springer-Verlag.

[107] T. Kivinen and M. Kojo, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)* RFC 3526, IETF.

[108] N. Koblitz, A. J. Menezes, *Another Look at "Provable Security"*, Journal of Cryptology, **20**:1 (2007), 3–27, Springer-Verlag.

[109] P. C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptography, Proceedings of CRYPTO 1996, LNCS **1109**, pp. 104–113, Springer-Verlag, 1996.

[110] P. C. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, Advances in Cryptography, Proceedings of CRYPTO 1999, LNCS **1666**, pp. 388–397, Springer-Verlag, 1999.

[111] F. Koeune, G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-P. David and J.-D. Legat, *A FPGA Implementation of the Linear Cryptanalysis*, In *12th International Conference on Field Programmable Logic and Applications (FPL 2002), Montpellier, France.*

[112] H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104, available at http://www.ietf.org/rfc/rfc2104.txt?number=2104

[113] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler, *How to Break DES for €8,980*, Workshop record of SHARCS 2006, www.ruhr-uni-bochum.de/itsc/tanja/SHARCS/

[114] A. K. Lenstra, *Unbelievable security; matching AES security using public key systems,* Proceedings of Asiacrypt 2001, LNCS **2248**, pp. 67–86, Springer-Verlag, 2001.

[115] A. K. Lenstra, *Key Lengths*, Chapter 114, of *The Handbook of Information Security*, Wiley 2005.

[116] A. K. Lenstra, A. Shamir, J. Tomlinson, and E. Tromer, *Analysis of Bernstein's factorization circuit,* Proceedings of Asiacrypt 2002, LNCS **2501**, pp. 1–26, Springer-Verlag, 2002.

[117] A. K. Lenstra and E. R. Verheul, *Selecting Cryptographic Key Sizes*, Journal of Cryptology **14**:4, 255–293, 2001.

[118] A. K. Lenstra and B. de Weger, On the Possibility of Constructing Meaningful Hash Collisions for Public Keys, Proceedings of ACISP 2005, LNCS **3574**, pp. 267–279, Springer-Verlag, 2005.

[119] A. K. Lenstra, X. Wang, and B. de Weger, *Colliding X.509 Certificates based on MD5-collisions*, http://www.win.tue.nl/~bdeweger/CollidingCertificates/

[120] G. Leurent, *Message Freedom in MD4 and MD5 Collisions: Application to APOP*, Proceedings of FSE 2007, LNCS **4593**, pp. 309–328, Springer-Verlag.

[121] C. McDonald, P. Hawkes and J. Pieprzyk. *SHA-1 collisions now* $2^{52}$. Eurocrypt 2009 Rump session, `http://eurocrypt2009rump.cr.yp.to/837a0a8086fa6ca714249409ddfae43d.pdf`.

[122] S. Maitra and G. Paul, *New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4*, Proceedings of FSE 2008, LNCS **5086**, pp. 250–266, Springer-Verlag, 2008.

[123] J. Manger, *A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1 v2.0*, Proceedings of Crypto 2001, LNCS **2139**, pp. 230–238, Springer-Verlag.

[124] M. Matsui, *Linear cryptanalysis method for DES cipher*, Proceedings of EUROCRYPT 93, LNCS **765**, pp. 386–397, Springer-Verlag.

[125] K. Matusiewicz, J. Pieprzyk, N. Pramstaller, C. Rechberger, and V. Rijmen, *Analysis of simplified variants of SHA-256*, Proceedings of WEWoRC 2005, LNI P-74, pp. 123–134, 2005.

[126] U. Maurer, *Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms*, Proceedings of CRYPTO '94, LNCS **839**, pp. 271–281, Springer-Verlag.

[127] U. Maurer and S. Wolf, *Diffie-Hellman, Decision Diffie-Hellman, and Discrete Logarithms*, Proceedings of ISIT '98, IEEE Information Theory Society, pp. 327, 1998.

[128] U. Maurer and S. Wolf, *The Relationship Between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms*, SIAM J. Comp., **28**:5 (1999), 1689–1721.

[129] A. Maximov and T. Johansson, *Fast Computation of Large Distributions and Its Cryptographic Applications*, Proceedings of Asiacrypt 2005, LNCS **3788**, pp. 313–332, Springer-Verlag, 2005.

[130] A. Maximov and D. Khovratovich, *New State Recovery Attack on RC4*, Proceedings of Crypto 2008, LNCS **5157**, pp. 297–316, Springer-Verlag.

[131] D. A. McGrew and S. R. Fluhrer, *Attacks on Additive Encryption of Redundant Plaintext and Implications on Internet Security*, Proceedings of SAC 2000, LNCS **2012**, pp. 14–24, Springer-Verlag, 2001.

[132] McLoone et al., *High Performance Single-Chip FPGA Rijndael Algorithm Implementations*, In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2001, Paris, France, 2001*, LNCS.

[133] F. Mendel, N. Pramstaller, C. Rechberger, and V. Rijmen, *Analysis of Step-Reduced SHA-256*, Proceedings of FSE 2006, LNCS **4047**, pp. 126–143, Springer-Verlag.

[134] F. Mendel, N. Pramstaller and C. Rechberger, *Improved Collision Attack on the Hash Function Proposed at PKC'98*, Proceedings of ICISC 2006, LNCS **4296**, pp. 8–21, Springer-Verlag, 2006.

[135] F. Mendel, C. Rechberger and V. Rijmen, *Update on SHA-1*, Presented at Rump Session of CRYPTO 2007.

[136] F. Mendel, C. Rechberger, M. Schläffer and S.S. Thomsen. *The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl*, Proceedings of FSE 2006, LNCS **5665**, pp. 260–276, Springer-Verlag, 2006.

[137] A. Menezes, P. C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[138] C. J. Mitchell, K. G. Paterson, and A. Yau, *Padding Oracle Attacks on the CBC-mode encryption with random and secret IVs*, Proceedings of Fast Software Encryption (FSE) 2005, LNCS **3557**, pp. 308–329.

[139] C. J. Mitchell and V. Varadharajan, *Modified forms of cipher block chaining*, Computers and Security 10, pp. 37–40, 1991.

[140] NIST, *Data encryption standard (DES)*, FIPS PUB 46-3, available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[141] NIST, *Secure hash standard*, FIPS PUB 180-2, available at http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[142] NIST, *Digital Signature Standard (DSS)*, FIPS PUB 186-2, Available at http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[143] NIST, *Advanced Encryption Standard*, FIPS PUB 197, available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[144] NIST, *Recommendation for Block Cipher Modes of Operation*, SP 800-38, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[145] NIST, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, SP 800-38B, http://csrc.nist.gov/publications/nistpubs/800-38b/sp800-38b.pdf

[146] NIST, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, SP 800-38D, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[147] NIST, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, SP 800-38C, http://csrc.nist.gov/publications/nistpubs/800-38c/sp800-38c.pdf

[148] NIST, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, SP 800-67, http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf

[149] NIST, *Recommendation for Key Management — Part 1: General* SP 800-57, May 2006, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

[150] NIST, *Cryptographic Algorithms and Key Sizes for Personal Identity Verification*, April 2005, available via csrc.nist.gov/publications/nistpubs/800-78/sp800-78-final.pdf

[151] NIST, *PlutoPlus: An IKE Reference Implementation for Linux*, available at http://ipsec-wit.antd.nist.gov/newipsecdoc/pluto.html

[152] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, *Report on the Development of the Advanced Encryption Standard (AES)*, available at http://csrc.nist.gov/CryptoToolkit/aes/round2/r2report.pdf

[153] P. Q. Nguyen and I. Shparlinski, *The insecurity of the digital signature algorithm with partially known nonces*, Journal of Cryptology, **15**, 151–176, 2002. Also available at ftp://ftp.ens.fr/pub/dmi/users/pnguyen/PubDSA.ps.gz.

[154] P. Q. Nguyen and I. Shparlinski, *The insecurity of the elliptic curve digital signature algorithm with partially known nonces*, Design, Codes and Cryptography, 2002. Also available at ftp://ftp.ens.fr/pub/dmi/users/pnguyen/PubECDSA.ps.gz.

[155] NESSIE consortium, *Portfolio of recommended cryptographic primitives,* Feb. 2003, available via http://www.cryptonessie.org/

[156] NESSIE consortium, *NESSIE Security report*, available at https://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/D20-v2.pdf

[157] National Security Agency. *NSA Suite B Cryptography.* `http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`.

[158] K. Nyberg and J. Wallén, *Improved Linear Distinguishers for SNOW 2.0*, Proceedings of FSE 2006, LNCS **4047**, pp. 144–162, Springer-Verlag, 2006.

[159] P. C. van Oorschot and M. J. Wiener, *Parallel Collision Search with Cryptanalytic Applications,* Journal of Cryptology **12**:1 (1999), 1–28.

[160] H. Orman, *The Oakley Key Determination Protocol* RFC 2412, IETF.

[161] H. Orman and P. Hoffman, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys,* IETF RFC 3766/BCP 86, April 2004.

[162] D. A. Osvik, A. Shamir, E. Tromer, *Cache Attacks and Countermeasures: The Case of AES*, Proceedings of CT-RSA 2006, LNCS **3860**, pp. 1–20, Springer, 2006.

[163] D. Page, *Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel*, Technical report CSTR-02-003, Department of Computer Science, University of Bristol. June 2002. Available online at *http://www.cs.bris.ac.uk/Publications/pub_master.jsp?id=1000625*

[164] RSA Labs, *PKCS# 1: RSA Cryptography Standard*, available at http://www.rsasecurity.com/rsalabs/node.asp?id=2125

[165] B. Preneel and P. C. van Oorschot, *A key recovery attack on the ANSI X9.19 retail MAC*, Electronics Letters, **32**:17 (1996), 1568–1569. Available at http://www.scs.carleton.ca/~paulv/papers/pubs.html

[166] C. Rechberger and V. Rijmen, *On Authentication Using HMAC and Non-Random Properties*, Proceedings of Financial Cryptography 2007, LNCS **4886**, pp. 119–133, Springer-Verlag.

[167] C. Rechberger and V. Rijmen, *New Results on NMAC/HMAC when Instantiated with Popular Hash Functions*, Journal of Universal Computer Science (JUCS), Special Issue on Cryptography in Computer System Security, **14**:3, 2008, 347–376.

[168] V. Rijmen, *Cryptanalysis and design of iterated block ciphers*, PhD thesis, October 1997.

[169] RIPEMD, http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html

[170] R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, available at http://www.ietf.org/rfc/rfc1321.txt?number=1321

[171] RSA Labs, *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths*, RSA Labs Bulletin #13, available at www.rsasecurity.com/rsalabs/

[172] SECG. *Standards for Efficient Cryptography Group.* SEC1: Elliptic Curve Cryptography version 2.0, `http://www.secg.org`.

[173] Saggese et al., *An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm*, In *FPL 2003*, LNCS **2778**, pp. 292–302, Springer-Verlag.

[174] Y. Sasaki and K. Aoki. *Finding Preimages in Full MD5 Faster Than Exhaustive Search.* Proceedings of EuroCrypt 2009, LNCS **5479**, pp. 134–152, Springer-Verlag.

[175] Y. Sasaki, L. Wang, K. Ohta, N. Kunihiro, *Security of MD5 Challenge and Response: Extension of APOP Password Recovery Attack*, Proceedings of CT-RSA 2008, LNCS **4964**, pp. 1–18, Springer-Verlag.

[176] D. Schmidt, *On the Key Schedule of Blowfish*, Manuscript 2005, available at http://eprint.iacr.org/2005/063.

[177] B. Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, Proceedings of FSE 1994, LNCS **1008**, pp. 191–204, Springer-Verlag, 1994.

[178] A. Shamir, *Factoring Large Numbers with the TWINKLE Device (Extended Abstract),* Manuscript, 2000.

[179] A. Shamir and E. Tromer, *Factoring large numbers with the TWIRL device,* Proceedings of Crypto 2003, LNCS **2729**, pp. 1–26, Springer-Verlag, 2003.

[180] R. Shipsey, *How long. . . ?,*
NESSIE Report NES/DOC/RHU/WP3/015/a, available via
https://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase1/rhuwp3-015.pdf

[181] P. W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,* SIAM J. Sci. Statist. Comput., **26** (1997).

[182] V. Shoup, *A proposal for an ISO standard for public key encryption*, Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/

[183] R. D. Silverman, *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths,* RSA Laboratories Bulletin #13, April 2000.

[184] E. Skoudis and L. Zeltser, *Malware: Fighting Malicious Code,* Prentice Hall, 2003.

[185] N. P. Smart, *How Secure are elliptic curves over composite extension fields?*, Proceedings of Eurocrypt '01, LNCS **2045**, pp. 30–39. Springer-Verlag, 2001.

[186] JH. Song, R. Poovendran, J. Lee, *The AES-CMAC-96 Algorithm and Its Use with IPsec*, RFC4494, IETF, 2006.

[187] F. X. Standaert, *Secure and Efficient Use of Reconfigurable Hardware Devices in Symmetric Cryptography*, Ph. D. thesis, Faculté des sciences appliquées, Université catholique de Louvain.

[188] Standaert et al., *Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs*, In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2003, Cologne, Germany, 2003*, LNCS **2779**, pp. 334–350.

[189] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart, *Flaws in Applying Proof Methodologies to Signature Schemes*, Proceedings Crypto 2002, LNCS **2442**, pp. 93–110, Springer-Verlag. Also available at http://www.di.ens.fr/~pointche/pub.php?reference=MaPoSmSt02

[190] M. Stevens, A. K. Lenstra and B. de Weger, *Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities*, Proceedings of EUROCRYPT 2007, LNCS **4515**, pp. 1–22, Springer-Verlag.

[191] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D.Molnar, D.A. Osvik and B. de Weger. *Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate.* Proceedings of Crypto 2009, LNCS **5677**, Springer-Verlag.

[192] I. Tuomi, *The Lives and Death of Moore's Law,* Available via http://www.firstmonday.dk/issues/issue7_11/tuomi/

[193] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance,* Nature 414 (2001), 883–887.

[194] S. Vaudenay, *Hidden collisions on DSS*, Proceedings of Crypto'96, LNCS **1109**, pp. 83–88, Springer-Verlag, 1996.

[195] S. Vaudenay, *The Security of DSA and ECDSA*, Proceedings of PKC'03, LNCS **2567**, pp. 309–323 Springer-Verlag, 2003.

[196] S. Vaudenay, *Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC,...* Proceedings of Eurocrypt'02, LNCS **2332**, pp. 534–545, Springer-Verlag, 2002.

[197] S. Vaudenay, *On the weak keys of Blowfish*, Proceedings of FSE'96, LNCS **1039**, pp. 27–32, Springer-Verlag, 1996.

[198] S. Vaudenay and M. Vuagnoux. *Passive-Only Key Recovery Attacks on RC4*, Proceedings of SAC 2007, LNCS **4876**, pp. 344-359, Springer-Verlag, 2007.

[199] VESA, *DisplayPort Specification.* Available at www.vesa.org

[200] L. Wang, K. Ohta and N. Kunihiro, *New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5*, Proceedings of Eurocrypt 2008, **4965**, pp. 237–253, Springer-Verlag.

[201] X. Wang and D. Feng, X. Lai, and H. Yu, *How to Break MD5 and other Hash Functions*, Proceedings of Eurocrypt'05, LNCS **3494**, pp. 19–35, Springer-Verlag, 2005.

[202] X. Wang, Y.L. Yin, and H. Yu, *Finding Collisions in the Full SHA-1*, Proceedings of Crypto'05, LNCS **3621**, pp. 17–36, Springer-Verlag, 2005.

[203] X. Wang, *New Collision search for SHA-1*, Manuscript, presented at rump session of Crypto'05.

[204] D. Watanabe, A. Biryukov, and C. De Canniére, *A Distinguishing Attack of SNOW 2.0 with Linear Masking Method*, Proceedings of SAC 2003, LNCS **3006**, pp. 222–233, Springer-Verlag, 2004.

[205] Weil descent page, http://www.cs.bris.ac.uk/~nigel/weil_descent.html

[206] B. Weis, *The Use of RSA Signatures within ESP and AH,* IETF draft http://www.ietf.org/internet-drafts/draft-ietf-msec-ipsec-signatures-03.txt, Nov 2004.

[207] M. J. Wiener, *Performance Comparison of Public-Key Cryptosystems,* RSA CryptoBytes **4**:1 (1998), 1–5.

[208] L. C. Williams, *A Discussion of the Importance of Key Length in Symmetric and Asymmetric Cryptography,* Available via http://www.giac.org/practical/gsec/Lorraine_Williams_GSEC.pdf

[209] Xilinx. Xilinx Press Release #03142. Available at `http://www.xilinx.com/prs_rls/silicon_spart/03142s3_pricing.htm`.

[210] H. Yoshida and A. Biryukov, *Analysis of a SHA-256 Variant*, Proceedings of SAC 2005, LNCS **3897**, Springer-Verlag, pp. 245–260.

# Appendix A

# Glossary

Abbreviations, most of which are explained in more detail in the documents.

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| AES | Advanced Encryption Standard |
| ANSI | American National Standards Institute |
| ASIC | Application-Specific Integrated Circuit |
| CCA | Chosen Ciphertext Attack |
| CDH | Computational Diffie-Hellman Assumption |
| CMA | Chosen Message Attack |
| CPU | Central Processing Unit |
| CRT | Chineese Remainder Theorem |
| DDH | Decisional Diffie-Hellman Assumption |
| DES | Data Encryption Standard |
| DH | Diffie-Hellman |
| DLOG | Discrete Logarithm |
| DSA | Digital Signature Algorithm |
| DSS | Digital Signature Standard |
| FPGA | Field Programmable Gate Array |
| EC | Elliptic Curve |
| ECC | Elliptic Curve Cryptography |
| EFF | Electronic Frontier Foundation |
| ETSI | European Telecommunications Standards Institute |
| FIPS | Federal Information Processing Standard |
| HW | Hardware |
| TLB | Translation Lookaside Buffer |

| | |
|---|---|
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Taskforce |
| IEC | International Electrotechnical Commission |
| IKE | Internet Key Exchange |
| ISO | International Standardization Organization |
| IP | Internet Protocol |
| IV | Initialization Value |
| KEM | Key Encapsulation Method |
| KDF | Key Derivation Function |
| MAC | Message Authentication Code |
| MD | Message Digest |
| MIME | Multipurpose Internet Mail Extensions |
| MIPS | Mega/Million Instructions Per Second |
| NESSIE | New European Schemes for Signatures, Integrity and Encryption |
| NFS | Number Field Sieve |
| NIST | National Institute of Standards and Technology |
| NIST SP | NIST Special Publication |
| OAEP | Optimal Asymmetric Encryption Padding |
| PK | Public Key |
| PKCS | Public Key Cryptography Standard |
| PRNG | Pseudo-random Number Generator |
| PSS | Probabilistic Signature Scheme |
| QS | Quadratic Sieve |
| RFC | Request For Comments (see www.ietf.org) |
| ROM | Random Oracle Model |
| RSA | Rivest-Shamir-Adleman cryptosystem |
| SHA | Secure Hash Algorithm |
| TLS | Transport Layer Security |
| UMTS | Universal Mobile Telecommunication System |
| WTLS | Wireless TLS |
| ZK | Zero Knowledge |