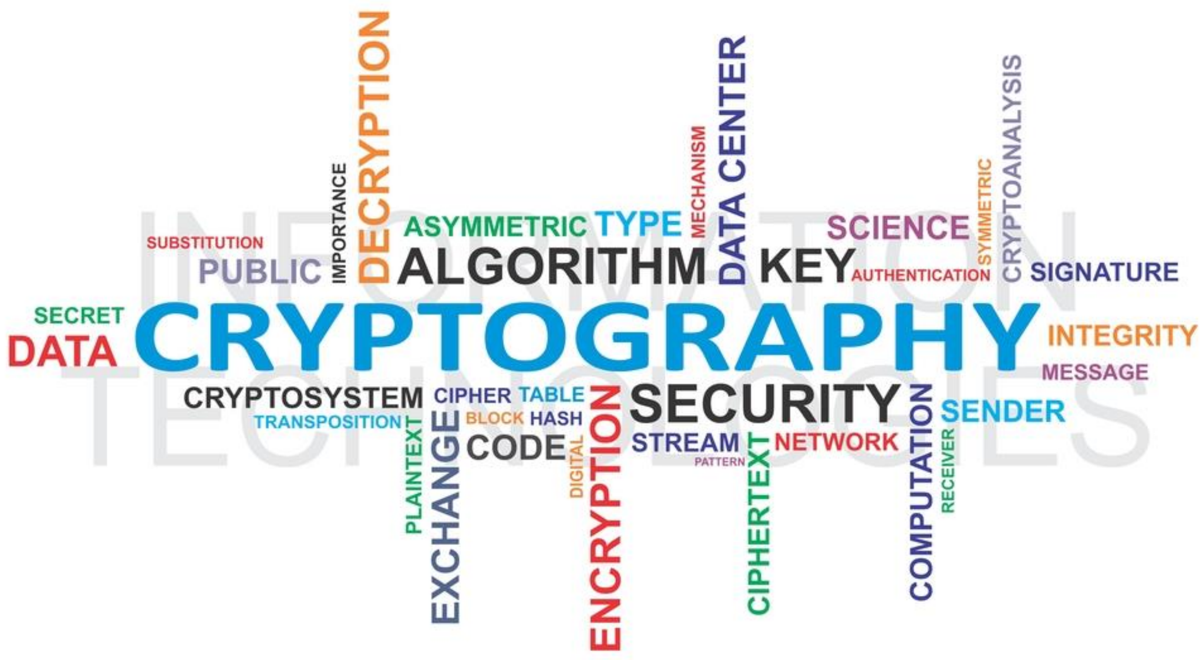

Algorithms, Key Sizes and Parameters Report

2013 recommendations

version 1.0 – October 2013

enisa

www.enisa.europa.eu

Contributors to this report:
This work was commissioned by ENISA under contract P/18/12/TCD Lot 2 to the consortium formed for this work by K.U.Leuven (BE) and University of Bristol (UK).

- Contributors: Nigel P. Smart (University of Bristol), Vincent Rijmen (K.U. Leuven), Bogdan Warinschi (University of Bristol), Gaven Watson (University of Bristol).
- Editor: Nigel P. Smart (University of Bristol),
- ENISA Project Manager: Rodica Tirtea

Agreements of Acknowledgements
We would like to extend our gratitude to:

- External Reviewers: Arjen Lenstra (EPFL), Christof Paar (Ruhr-Universität Bochum), Kenneth G. Paterson (Royal Holloway, University of London), Michael Ward (Mastercard) for their comments suggestions and feedback.
- We also thank a number of people for providing anonymous input.

About ENISA

Contact details

For contacting ENISA or for general enquiries on cryptography, please use the following details:

- E-mail: `sta@enisa.europa.eu`.
- Internet: `http://www.enisa.europa.eu`.

For questions related to current report, please use the following details:

- E-mail: `sta@enisa.europa.eu`

Legal Notice

Contents

Acronyms

3DES	Triple DES
3GPP	3rd Generation Partnership Project (mobile phone system)
A5/X	Stream ciphers used in mobile phone protocols
AES	Advanced Encryption Standard
AMAC	Ansi Retail MAC
BB	Boneh–Boyen (ID based encryption)
BF	Boneh–Franklin (ID based encryption)
CBC	Cipher Block Chaining (mode)
CCA	Chosen Ciphertext Attack
CCM	Counter with CBC-MAC (mode)
CFB	Cipher Feedback
CMA	Chosen Message Attack
CMAC	Cipher-based MAC
CPA	Chosen Plaintext Attack
CTR	Counter (mode)
CVA	Ciphertext Validity Attack
CWC	Carter–Wegman + Counter
DEM	Data Encapsulation Mechanism
DES	Data Encryption Standard
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
E0	A stream cipher used in Bluetooth
EAX	Actually stands for nothing (mode)
ECB	Electronic Code Book (mode)
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECIES	Elliptic Curve Integrated Encryption Scheme
EEA	EPS Encryption Algorithm
EIA	EPS Integrity Algorithm
EMAC	Encrypted CBC-MAC

EME	ECB-mask-ECB (mode)
EMV	Europay–Mastercard–Visa (chip-and-pin system)
ENISA	European Network and Information Security Agency
FDH	Full Domain Hash
GCM	Galois Counter Mode
GDSA	German Digital Signature Algorithm
GSM	Groupe Spécial Mobile (mobile phone system)
HMAC	A hash based MAC algorithm
IAPM	Integrity Aware Parallelizable Mode
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IND	Indistinguishability of Encryptions
INT-CTXT	Integrity of Ciphertexts
IPsec	Internet Protocol Security
ISO	International Standards Organization
IV	Initialisation Vector (or Value)
KDSA	Korean Digital Signature Algorithm
KDF	Key Derivation Function
KEM	Key Encapsulation Mechanism
LTE	Long Term Evolution (mobile phone system)
MAC	Message Authentication Code
MOV	Menezes–Okamoto–Vanstone (attack)
MQV	Menezes–Qu–Vanstone (protocol)
NIST	National Institute of Standards and Technology (US)
NMAC	Nested MAC
OAEP	Optimal Asymmetric Encryption Padding
OCB	Offset Code Book (mode)
OFB	Output Feedback (mode)

PKCS	Public Key Cryptography Standards
PRF	Pseudo Random Function
PRP	Pseudo Random Permutation
PSEC	Provable Secure Elliptic Curve (encryption)
PSS	Probabilistic Signature Scheme
PV	Pointcheval–Vaudenay (signatures)
RC4	Ron's Cipher Four (a stream cipher)
RDSA	Russian Digital Signature Algorithm
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
SK	Sakai–Kasahara (ID-based encryption)
SSH	Secure Shell
SSL	Secure Sockets Layer
UEA	UMTS Encryption Algorithm
UF	Universally Unforgeable
UIA	UMTS Integrity Algorithm
UMAC	Universal hashing based MAC
UMTS	Universal Mobile Telecommunications System
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing

Chapter 1

Executive Summary

Recently published EC Regulations on the measures applicable to the notification of personal data breaches [119] make reference to ENISA, as a consultative body, in the process of establishing a list of appropriate cryptographic protective measures. Furthermore, in a previous study of ENISA [113], addressing the use of cryptographic techniques in the EU, published in 2011 it had been noticed that a large number of national bodies were referring to the ECRYPT and ECRYPT II "Yearly Report on Algorithms and Key Lengths" [104–111]. As the ECRYPT Network of Excellence (NoE) published the last report at the end of 2012, the need for continuation of this activity has been acknowledged by ENISA in its work programme for 2013 [115].

This document collates a series of recommendations for algorithms, keysizes, and parameter recommendations. It addresses the need for a minimum level of requirements for cryptography across European Union (EU) Member States (MSs) in their effort to protect personal and sensitive data of the citizens. The document tries to address the need for continuation of the reports published by ECRYPT NoE and also the requirements for cryptographic protective measures applicable to the notification of personal data breaches by providing a focused set of recommendations in an easy to use form. This is the first report consisting of recomendations addressing cryptographic algorithms, sizes and paramenters published by ENISA. The intention is to continue and extend this activity in the following years.

It should be noted that this is a technical document addressed to decision makers, specialists designing and implementing cryptographic solutions. This set of recommendations is complementing another study [114] published by ENISA that provides an easy to read and understand context for non-speciallized parties and which places the notions of information security in the context of personal data protection framework.

In this document we focus on just two decisions which we feel are more crucial to users of cryptography.

Firstly, whether a given primitive, scheme or protocol can be considered for use today if it is already deployed. We refer to such use as *legacy* use within our document. Our first recommendation is that if a scheme is not considered suitable for legacy use, or is only considered for such use with certain caveats, then this should be taken as a strong recommendation that the primitive, scheme or protocol be replaced as a matter of urgency.

Secondly, we consider the issue of whether a primitive, scheme, or protocol is suitable for deployment in new or future systems. In some sense mechanisms which we consider useable for new and future systems meet cryptographic requirements described in this document; they generally will have proofs of security, will have key sizes equivalent to 128-bit symmetric security or more[1], will have no structural weaknesses, will have been well studied, will have been been standardized, and will have a reasonably-sized existing user base. Thus the second recommendation is that decision makers now make plans and preparations for the phasing out of what we term legacy mechanisms over a period of say 5-10 years, and replacing them with systems we deem secure for future use.

The document also details a summary of some of the many cryptographic protocols in use. This section is deliberately brief as there has been very little scientific analysis of cryptographic protocols in the literature. It is hoped that by detailing some protocols in need of analysis the research community will start to consider examining these in more detail.

This document does not consider any mechanisms which are currently only of academic interest. In particular all the mechanisms we discuss have been standardized to some extent, and have either been deployed, or are slated to be deployed, in real systems. This selection is a means of focusing the document on mechanisms which will be of interest to decision makers in industry and government.

As another restriction of scope, we do not consider issues related to implementation (e.g. side channels resulting from timing, power, cache analysis, etc), nor issues arising from poor randomness generation.

Further limitations of scope are mentioned in the introductory chapter which follows. Further restrictions are mentined in Chapter 2 "How to Read this Document". Such topics, which are not explored by this document, could however be covered in the future.

[1]See Section 3.6 for the equivalence mapping between symmetric key sizes and public key sizes

Chapter 2

How to Read this Document

This document collates a series of recommendations for algorithm, keysize and protocol recommendations. In some sense the current document supersedes the ECRYPT and ECRYPT2 "Yearly Report on Algorithms and Key Lengths" published between 2004 and 2012 [104–111]. However, it should be considered as completely distinct. The current document tries to provide a focused set of recommendations in an easy to use form, the prior ECRYPT documents provided more general background information and discussions on general concepts re key size choice, and tried to predict the future ability of cryptanalytic attacks via hardware and software.

In this document we focus on just two decisions which we feel are more crucial to users of cryptography. Firstly, whether a given primitive, scheme, protocol or keysize can be considered for use today if it is already deployed. We refer to such use as *legacy* use within our document. If a scheme is *not* considered suitable for legacy use, or is only considered for such use with certain caveats, then this should be taken as a *strong recommendation* that the primitive, scheme or protocol be possibly *replaced* as a matter of urgency (or even that an attack exists). Some of the caveats which may mean a system which it not considered suitable for legacy use may still be secure could be use of limited key lifetimes within a system, mitigating controls, or (in the case of hash functions) relieing on non-collision resistance properties.

In particular, we stress, that schemes deemed to be legacy are considered to be secure currently. But, that for future systems there are better choices available which means that retaining systems which we deem to be legacy in future systems is not best practice. We summarize this distinction in Table 2.1.

Secondly, we consider the issue of whether a primitive, scheme, protocol, or key size is suitable for deployment in new or future systems. In some sense mechanisms which we consider usable for new and future systems meet a gold standard of cryptographic strength; they generally will have proofs of security, will have key sizes equivalent to 128-bits symmetric security or more, will have no structural weaknesses, will have been well studied, been standardized and would have a reasonable existing install base.

Classification	Meaning
Legacy ✗	Attack exists or security considered not sufficient. Mechanism should be replaced in fielded products as a matter of urgency.
Legacy ✓	No known weaknesses at present. Better alternatives exist. Lack of security proof or limited key size.
Future ✓	Mechanism is well studied (often with security proof). Expected to remain secure in 10-50 year lifetime.

Table 2.1: Summary of distinction between legacy and future use

As a general rule of thumb we consider symmetric 80-bit security levels to be sufficient for legacy applications for the coming years, but consider 128-bit security levels to be the minimum requirement for new systems being deployed. Thus the key recommendation is that decision makers now make plans and preparations for the phasing out of what we term legacy mechanisms over a period of say 5-10 years. In selecting key sizes for future applications we consider 128-bit to be sufficient for all but the most sensitive applications. Thus we make no distinction between high-grade security and low-grade security, since 128-bit encryption is probably secure enough in the near term.

However, one needs to also take into account the length of time data needs to be kept secure for. For example it may well be appropriate to use 80-bit encryption into the near future for transactional data, i.e. data which only needs to be kept secret for a very short space of time; but to insist on 128-bit encryption for long lived data. All recommendations in this document need to be read with this in mind. We concentrate on recommendations which imply a minimal security level across all applications; i.e. the most conservative approach. Thus this does not imply that a specific application may enable lower security levels than that considered here.

The document does not consider any mechanisms which are currently only of *academic* interest. In particular all the mechanisms we discuss have been standardized to some extent, and have either been deployed or are due to be deployed in real world systems. This is not a critique of academic research, but purely a means of focusing the document on mechanisms which will be of interest to decision makers in industry and government.

As another restriction of scope we do not consider issues related to implementation (e.g. side channels resulting from timing, power, cache analysis etc), nor insufficient randomness generation [215]. However, we do consider implementation issues related to the mathematical instantiation of the scheme, such as padding oracle attacks etc. If a particular mechanism option provides a defense against side channel attacks (for example) we may mention it, but implementers are *strongly* recommended to use all the standard implementation techniques to defend against such attacks. In terms of randomness generation, almost all cryptographic mechanisms require access to a sufficient entropy source. Whilst out of scope of this document, implementors are again *strongly* recommended to ensure appropriate steps are taken to provide sufficient entropy to the mechanism.

As another restriction of scope, which we alluded to above, we do not make a comprehensive discussion on how key size equivalents are decided upon (e.g. what RSA key size corresponds to what AES key size). We refer to other comparisons in the literature in Section 3.1, but we feel repeating much of this analysis would detract from the focus of this document.

2.1 Division into Chapters and Sections

The document divides cryptographic mechanisms into primitives (such as block ciphers, public key primitive and hash functions), schemes (such as symmetric and public key encryption schemes, signature schemes etc), and protocols (such as key agreement, TLS, IPsec etc). Protocols are built out of schemes, and schemes are themselves built out of primitives. At each stage of this process security needs to be defined, and the protocol or scheme needs to be proven to meet this definition, given the components it uses. So for example, just because a scheme makes use of a secure primitive does not imply the scheme is secure; this needs to be demonstrated by a proof. Luckily for most schemes such proofs do exist. However, in the case of protocols very little work has been performed in proving that deployed protocols meet a well defined security definition even when they are built out of secure component primitives and schemes.

$$\begin{tikzcd}\text{Primitive} \arrow[r] & \text{Scheme} \arrow[r] & \text{Protocol}\end{tikzcd}$$

Cryptographic primtives are considered the basic building blocks upon which one needs to make some *assumption*. This assumption is the level of difficulty of breaking this precise building block; this assumption is always the communities current "best guess". We discuss primitives in detail in Chapter 3.

In Chapter 4 we then go onto discuss schemes. By a scheme we mean some method for taking a primitive, or set of primitives, and constructing a cryptographic service out of the primitive. Hence, a scheme could refer to a digital signature scheme or a mode of operation of a block cipher. It is considered good cryptographic practice to only use schemes for which there is a *well defined security proof* which reduces the security of the scheme to that of the primitive. So for example an attack against CBC mode using AES should result in an attack against the AES primitive itself.

Making the distinction between schemes and primitives also means we can present schemes as general as possible and then allow users to instantiate them with secure primitives. However, this leads to the question of what generally should the key size be for a primitive given, if is to be used within a scheme? This might seem a simple question, but it is one which divides the cryptographic community. There are two approaches to this problem:

1. Since a security proof which reduces security of a scheme to an underlying primitive can

introduce a *security loss*, some cryptographers state that the key size of the primitive should be chosen with respect to this loss. With such a decision, unless proofs are *tight*[1], the key sizes used in practice will be larger than one would normally expect. The best one can hope for is that the key size for the scheme matches that of the underlying primitive.

2. Another school of thought says that a proof is just a design validation, and the fact a tight proof does not exist may not be for fundamental reasons but could be because our proof techniques are not sufficiently advanced. They therefore suggest picking key sizes to just ensure the underlying primitive is secure.

It is this second, pragmatic, approach which we adopt in this document. It is also the approach commonly taken in industry.

The third type of cryptographic construction we examine is that of protocols (Chapter 5). Whilst the design of primitives, and the analysis of schemes via provable security, has reached a high level of sophistication; the analysis of protocols lags far behind. The academic literature does contain analysis of quite complex protocols, but often these are designed from scratch to enable efficient analysis. There is little analysis of existing protocols, in which a cryptographic analysis needs to be performed after the design has been deployed or fixed. Indeed, many real world protocols do not withstand such rigorous analysis and numerous weaknesses have been discovered in real world protocols due to a lack of rigorous analysis before they were deployed. In an ideal world design and analysis should be completed before deployment. However, in recent years a number of researchers have started to examine how to do a post-hoc security analysis on already defined protocols.

In the chapter on protocols we present, for a number of existing well known protocols, the level of cryptographic security analysis which is known to have been performed. We restrict in this section, purely for reasons of space, to those results of a cryptographic nature; for example padding/error oracle attacks which we consider to be cryptographic design problems.

2.2 Making a Decision

The question then arises as to how to read this document? Whilst the order of the document is one of going from the ground up, the actual order of making a decision should be from the top down. We consider two hypothetical situations. One in which a user wishes to select a public key signature algorithm and another in which he wishes to select a public key encryption algorithm for use in a specific protocol. Let us not worry too much about which protocol is being used, but assume that the protocol says that one can select either RSA-PSS or EC-Schnorr as the public key signature algorithm, and either RSA-OAEP or ECIES as the public key encryption algorithm.

[1] i.e. there is no noticeable security loss in the proof

2.2.1 Public key signatures

We first examine the signature algorithm case. The reader should first turn to the section on signature schemes in Section 4.8. The reader should examine the discussion of both RSA-PSS and EC-Schnorr in Sections 4.8.2 and 4.8.7 respectively. One finds that both signature schemes are considered suitable for legacy applications and future applications. However, for "systems" reasons (probably the prevalence of RSA based digital certificates) the user decides to go for RSA-PSS. The RSA-PSS scheme is actually made up of two primitives; firstly the RSA primitive (discussed in Section 3.5.1) and secondly a hash function primitive (discussed in Section 3.3). Thus the user now needs to consider "which" RSA primitive to use (i.e. the underlying RSA key size) and which hash function to use. The scheme itself will impose some conditions on the relevant sizes so they match up, but this need not concern a reader of this document in most cases. Returning to RSA-PSS we see that the user should use 1024-bit RSA moduli only for legacy applications and SHA-1 as a hash function only for legacy applications. If that is all the user requires then this document would support the user's decision. However, if the user is looking at creating a new system without any legacy concerns then this document cannot be used as a justification for using RSA moduli of 1024 bits and SHA-1 as the hash function. The user would instead be forced to consider RSA moduli of 3024 bits (or more) and a hash function such as the 256-bit variant of SHA-2.

2.2.2 Public key encryption

We now turn to comparing the choice of RSA-OAEP and the ECIES hybrid cipher. By examining Chapter 4 on schemes (in particular Section 4.6.2 for RSA-OAEP and Section 4.7 for ECIES) the user sees that whilst both schemes have security proofs and so can be used for future applications, ECIES is better suited to long messages. They therefore decide to proceed with ECIES, which means certain choices need to be made with respect to the various components. The ECIES public key encryption scheme, being a hybrid cipher, is made from the ECIES-KEM scheme (see Section 4.7.3), which itself makes use of a key derivation method (see Section 4.4 for various choices of key derivation methods) and a Data Encapsulation Method, or DEM. A DEM is a form of one-time authenticated symmetric encryption, see Section 4.3 for various possible instantiations. This creates a huge range of possible instantiations, for which we now outline a possible decision process and which we illustrate graphically in Figure 2.1. From examining Section 4.7.3 on ECIES-KEM and Section 4.3 on authenticated symmetric encryption the user sees that ECIES-KEM is supported for legacy and future use, and that so is Encrypt-Then-MAC as a DEM. Given these choices for the components the user then needs to instantiate Encrypt-Then-MAC, which requires the choice of an IND-CPA symmetric encryption scheme (i.e. a block cipher mode of operation from Section 4.1) and a MAC algorithm from Section 4.2. Looking at these sections the user then selects CTR mode (for use with some block cipher), and CMAC (again for use with some block cipher). The KEM also requires use of a key derivation function from Section 4.4, which will output a key for the block cipher in CTR mode and a separate key for the CMAC algorithm. The user at this point could select the key derivation function that we denote X9.63-KDF, which itself requires the use of a hash function. Only at this point does the user of this document examine Chapter 3 on primitives so as to instantiate the precise elliptic curve group, the precise hash function for use in the key derivation function and the block ciphers to be used in the CTR mode encryption and the CMAC function. At this point a valid choice (for future applications) could be a 256-bit elliptic curve group, the SHA-2 key derivation function, and the AES block cipher at 128-bit key-length.

We stress that the above decision, on how to instantiate ECIES, is just one possible amongst all the various methods which this document supports.

2.3 Comparison to Other Documents

This document is one of many which presents recommendations for cryptographic primitives, key sizes and schemes. Each of these documents has a different audience and purpose; our goal has been to present an analysis of algorithms commonly used in current practice as well as providing state-of-the-art advice as to adoption of algorithms in future systems. Our recommendations are often rather conservative since we aim to give recommendations for the constructions of systems with a long live cycle.

As already remarked there is a strong relationship between this document and the ECRYPT and ECRYPT2 reports [104–111]. As mentioned earlier the current document is focused on making explicit recommendations as opposed to providing a general framework and summary as the original ECRYPT documents did.

Various government organizations provide advice, see annex A of [113], or mandates, in relation to key size and algorithm choice for their own internal purposes. In these documents, the choice of algorithms and key sizes is often done with an eye to internal systems and processes. The current document extends the scope to a wider area, e.g., internet communication and hence in addition considers algorithms deployed in various internet protocols.

Among the EU member states, there are a number of such documents including [21] published by France, and [66, 67] published by Germany. The key size recommendations of these three documents are in almost all cases consistent with our own recommendations for symmetric key sizes, hash function sizes and elliptic curve key sizes. The documents [66] and [21] also mention integer factorization based primitives; the recommendations in [21] are consistent with our own, whilst we are more conservative than [66] in this respect. Along with [21] we place a strong emphasis on using schemes with modern security proofs.

Further afield the US government maintains a similar document called Suite B [251], which presents recommended algorithms and key sizes for various govenmental uses. Again our recommendations are broadly consistent in terms of key sizes with this document.

All of these documents [21, 66, 67, 251] also detail a number of concrete cryptographic schemes. In this aspect our coverage is much wider due to our wider audience. For example all documents recommend the use of AES, SHA-2 and elliptic curve based primitves, and some integer factorization based primitives. As well as these basic primitives we also mention a number of other primitives which are used in various deployed protocols, for example Camellia (in TLS), SNOW 3G (in GSM/LTE), as well as primitives used in what we term legacy systems (e.g. MD5, SHA-1, DES etc).

In terms of cryptographic schemes our coverage is much wider than that of [66, 67, 251]; this is only to be expected as per our different audiences. As an example of this we cover a significant number of MAC functions, authenticated encryption modes, and key derivation functions compared to the other documents. In one aspect we diverge from [66,67,251] in that we recommend variants of the DSA algorithm for use in legacy systems only. This is because DSA only has a security proof in a relatively weak computational model [63]. For discrete logarithm based signatures we recommend Schnorr signatures [313], which have stronger provable security properties than DSA [254, 283].

Another form of comparison can be made with the documents of various standards organizations. The ones which have been most referred to in this report are those of IETF, ISO and NIST. Divergences from the recommendations (if any) in these standards are again due to the distinct audiences. The IETF standardizes the protocols which keeps the internet running, their main concern is hence interoperability. As we have seen in recent months, with attacks on TLS and IPSec, this often leads to compromises in algorithm selection and choice. The ISO takes a very liberal approach to standardizing cryptographic algorithms, with far more algorithms standardized than a report like this could reasonably cover. We have selected algorithms from ISO (and dubbed them suitable/unsuitable for legacy and future use) due to our perception of their importance in other applications. Finally the NIST documents are more focused, with only a small subset of schemes being standardized. A major benefit in the NIST standardization is that when security proofs are available they are alluded to, and so one can judge the scientific basis of the recommendations.

2.4 Open Issues and Areas Not Covered

Whilst the area of primitives and schemes has attracted considerable academic attention in the last three decades, the same cannot be said of high level protocols. Whilst there are some academic treatments of protocols (e.g. extensive work on key agreement), very few of the actual deployed protocols make use of the design methodology introduced in the academic treatments. Thus for protocols there is a wide divergence between cryptographic theory and cryptographic practice. We give an overview of some (well used) protocols in this document which have had some limited analysis. A number of researchers are now trying to bridge this divide with considerable progress having been made in the last few years; but more work needs to be urgently done in this area.

Mainy of the recommendations in this document are focused on long term data retention issues (e.g. encrypted stored data, or long term signatures). Many cryptographic systems only need to protect transient data (i.e. transactional data) which has no long term value. In such situations some of the recommendations with respect to key size etc may need to be changed.

Due to time constraints there are also a number of areas which we have not touched upon in this document. In terms of cryptographic schemes these contain, but are not limited to:

- Currently practical Post-Quantum Systems.
- Random number generation (e.g. physical generation) and extraction (via pseudo-random number generators), e.g. as in FIPS 140-2 or NIST SP 800-90.
- Idenfitication and User Authentication Protocols, e.g. as in the various ISO 9798 standards.
- Key-wrap algorithms, e.g. as in ANSI X9.102 or NIST SP 800-38F.
- Key life cycle management, e.g. as in NIST SP 800-130 and 800-132,
- Password based key derivation, e.g. as in NIST SP 800-132.
- Password based key agreement protocols, e.g. EKE, PAK, SPEKE and J-PAKE.

It is hoped that if this document were to be revised in future years that the opportunity would be taken to also include the afore mentioned mechanisms.

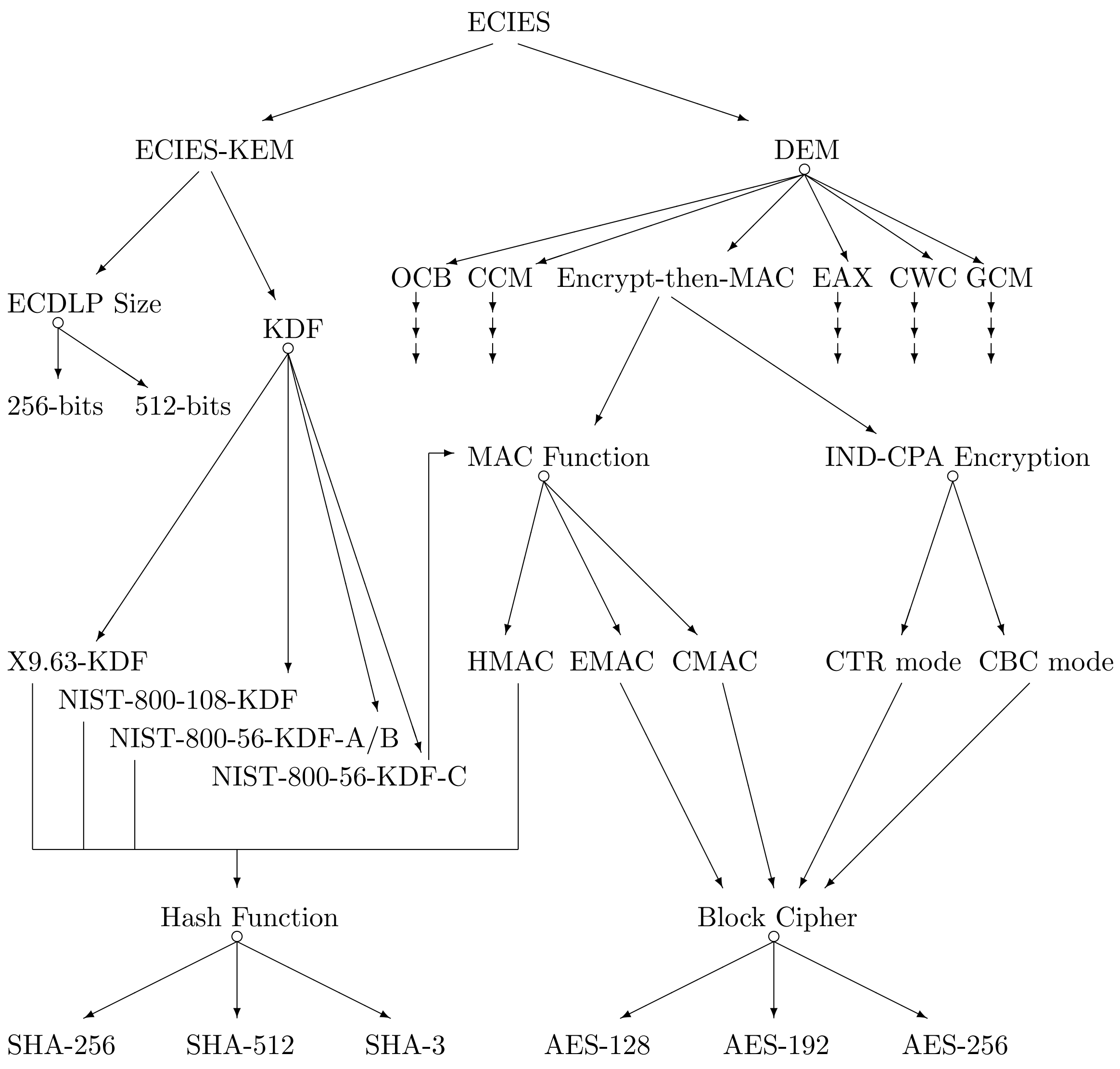

Figure 2.1: Just some of the design space for instantiating the ECIES public key encryption algorithm. Note, that not all standards documents will support all of these options. To read this diagram: A group of arrows starting with a circle implies the implementer needs to choose one of the resulting paths. A set of three arrows implies a part of the decision tree which we have removed due to space. In addition (again for reasons of space) we do not list all possible choices. Even with these restrictions one can see the design space for a cipher as well studied and understood as ECIES can be quite complex.

Chapter 3

Primitives

This chapter is about basic cryptographic building blocks, the atoms out of which all other cryptographic constructions are produced. In this section we include basic symmetric key building blocks, such as block ciphers, hash functions and stream ciphers; as well as basic public key building blocks such as factoring, discrete logarithms and pairings. With each of these building blocks there is some mathematical hard problem underlying the primitive. For example the RSA primitive is based on the difficulty of factoring, the AES primitive is (usually) based on it representing a keyed pseudo-random permutation. That these problems are hard, or equivalently, the primitives are secure is an assumption which needs to be made. This assumption is often based on the specific parameters, or key lengths, used to instantiate the primitives.

Modern cryptography then takes these building blocks/primitives and produces cryptographic schemes out of them. The defacto methodology, in modern work, is to then show that the resulting scheme, when attacked in a specific cryptographic model, is secure assuming the underlying assumption on the primitive holds. So another way of looking at this chapter and the next, is that this chapter presents the constructions for which we cannot prove anything rigorously, whereas the next chapter presents the schemes which should have proofs relative to the primitives in this chapter actually being secure.

In each section we use the term *observation* to point out something which may point to a longer term weakness, or is purely of academic interest, but which is not a practical attack at the time of writing. In each section we also give a table, and group the schemes within the table in order of security recommendations (usually).

3.1 Comparison

In making a decision as to which cryptographic mechanism to employ, one first needs to decide the mechanism and then decide the key length to be used. In later sections and chapters we focus on the mechanism choice, whereas in this section we focus just on the key size. In some schemes the effective key length is hardwired into the mechanism, in others it is a parameter to be chosen, in some there are multiple parameters which affect the effective key length.

There is common understanding that what we mean by an effective key length is that an attack should take 2^k operations for an effective key length of k. Of course this understanding is itself not well defined as we have not defined what an operation is; but as a rule of thumb it should be the "basic" operation of the mechanism. This lack of definition of what is meant by an operation means that it is hard to compare one mechanism against another. For example the best attack against a block cipher of key length k_b should be equivalent to 2^{k_b} block cipher invocations, whereas the best known attack against an elliptic curve system with group order of k_e bits should be $2^{k_e/2}$ elliptic curve group operations. This often leads one to conclude that one should take $k_e = 2 \cdot k_b$, but this assumes that a block cipher call is about the same cost as an elliptic curve group operation (which may be true on one machine, but not true on another).

This has led authors and standards bodies to conduct a series of studies as to how key sizes should be compared across various mechanisms. The "standard" method is to equate an effective key size with a specific block cipher, (say 112 corresponds to two or three key Triple-DES, 128 corresponds to AES-128, 192 corresponds to AES-192, and 256 corresponds to AES-256), and then try to establish an estimate for another mechanisms key size which equates to this specific quanta of effective key size.

In comparing the different literature one meets a major problem in that not all studies compare the same base symmetric key sizes; or even do an explicit comparison. The website `http://www.keylength.com` takes the various proposed models from the the literature and presents a mechanism to produce such a concrete comparison. In Table 3.1 we present either the concrete recommendations to be found in the literature, or the inferred recommendations presented on the web site `http://www.keylength.com`.

We focus on the symmetric key size k, the RSA modulus size $\ell(N)$ (which is also the size of a finite field for DLP systems) and the discrete logarithm subgroup size $\ell(q)$; all of which are measured in bits. Of course these are just crude approximations and hide many relationships between parameters which we discuss in future sections below. As one can see from the table the main divergence in estimates is in the selection of the size $\ell(N)$ of the RSA modulus.

As one can see, as the symmetric key size increases the size of the associated RSA moduli needs to become prohibitively large. Ignoring such large value RSA moduli we see that there is surprising agreement in the associated size of the discrete logarithm subgroup q, which we assume to be an elliptic curve group order.

Our implicit assumption is that the above key sizes are for (essentially) single use applications. As a key is used over and over again its security degrades, due to various time-memory tradeoffs. There are often protocol and scheme level procedures to address this issue; for example salting in password hashing. The same holds true in other situations, for example in [46], it is shown that AES-128 has only 85-bit security if 2^{43} encryptions of an arbitrary fixed text under different keys are available to the attacker.

Very little literature discusses the equivalent block length for block ciphers or the output length

k	$\ell(N)$	$\ell(q)$	k	$\ell(N)$	$\ell(q)$	k	$\ell(N)$	$\ell(q)$	k	$\ell(N)$	$\ell(q)$	k	$\ell(N)$	$\ell(q)$
Lenstra–Verheul 2000 [217] $\star$														
80	1184	142	112	3808	200	128	5888	230	192	20160	350	256	46752	474
Lenstra 2004 [214] $\star$														
80	1329	160	112	3154	224	128	4440	256	192	12548	384	256	26268	512
IETF 2004 [270] $\star$														
80	1233	148	112	2448	210	128	3253	242	192	7976	367	256	15489	494
SECG 2009 [314]														
80	1024	160	112	2048	224	128	3072	256	192	7680	384	256	15360	512
NIST 2012 [265]														
80	1024	160	112	2048	224	128	3072	256	192	7680	384	256	15360	512
ECRYPT2 2012 [107]														
80	1248	160	112	2432	224	128	3248	256	192	7936	384	256	15424	512

Table 3.1: Key Size Comparisons in Literature. An entry marked with a $\star$ indicates an inferred comparison induced from the web site `http://www.keylength.com`. Where a range is given by the source we present the minimum values. In the columns k is the symmetric key size, $\ell(N)$ is the RSA modulus size (or finite field size for finite field discrete logarithms) and $\ell(q)$ is the subgroup size for finite field and elliptic curve discrete logarithms.

of hash functions or MAC functions; since this is very much scheme/protocol specific. A good rule of thumb for hash function outputs is that they should correspond in length to $2 \cdot k$, since often hash functions need to be collision resistant. However, if only preimage or second-preimage resistance is needed then output sizes of k can be tolerated.

The standard [262] implicitly recommends that the MAC key and and MAC output size should be equal to the underlying symmetric key size k. However, the work of Preneel and van Oorschot [287, 288], implies attacks on MAC functions requiring $2^{n/2}$ operations, where n is the key size, or the size of the MAC functions internal memory. These recommendations can be problematic with some MAC function constructions based on block ciphers at high security levels, as no major block cipher has block length of 256 bits. In addition one needs to distinguish between off-line attacks, in which a large MAC output size is probably justified, and an on-line attack, where smaller MAC output sizes can be tolerated. Thus choice of the MAC output size can be very much scheme, protocol, or even system, dependent.

3.2 Block Ciphers

By a block cipher we mean (essentially) a keyed pseudo-random permutation on a block of data of a given length. A block cipher is *not* an encryption scheme, it is a component (in our terminology *primitive*) which goes into making such a scheme; often this is done via a mode of operation. In this section we consider whether a given block cipher construction is secure, in the sense that it seems to act like a pseudo-random permutation. Such a security consideration can never be proven, it is a mathematical assumption, akin to the statement that factoring 3024-bit moduli is hard. The schemes we present in Chapter 4, that use block ciphers, are often built on the assumption that the block cipher is secure in the above sense.

Generally speaking we feel the minimum key size for a block cipher should be 128 bits; the minimum for the block size depends on the precise application but in many applications (for example construction of MAC functions) a 128-bit block size should now be considered the minimum in many application. We also consider that the maximum amount of data which should be encrypted under the same key should be bounded by $2^{n/2}$, where n is the block size in bits. However, as indicated before some short lived cryptograms may warrant smaller block and key sizes in their constructions; but for general applications we recommend a minimum of 128 bits.

Again, for each primitive we give a short description of state of the art with respect to known attacks, we then give recommendations for minimum parameter sizes for future and legacy use. For convenience these recommendations are summarized in Table 3.2.

	Recommendation	
Primitive	Legacy	Future
AES	✓	✓
Camellia	✓	✓
Three-Key-3DES	✓	✗
Two-Key-3DES	✓	✗
Kasumi	✓	✗
Blowfish$^{\geq 80\text{-bit keys}}$	✓	✗
DES	✗	✗

Table 3.2: Block Cipher Summary

3.2.1 Recommended Block Ciphers

AES

The Advanced Encryption Standard, or AES, is the block cipher of choice for future applications [88, 120]. AES is called 128-EIA 2 in LTE. The AES has a block length of 128 bits and supports 3 key lengths: 128, 192 and 256 bits. The versions with longer key lengths use more rounds and are hence slower (by 20, respectively 40%).

Observation: The strong algebraic structure of the AES cipher has led some researchers to suggest that it might be susceptible to algebraic attacks [86, 249]. However, such attacks have not been shown to be effective [76, 220].

For the 192- and 256-bit key versions there are related key attacks [44, 45]. For AES-256 this attack, using four related keys, requires time $2^{99.5}$ and data complexity $2^{99.5}$. The attack works due to the way the key schedule is implemented for the 192- and 256-bit keys (due to the mismatch in block and key size), and does not affect the security of the 128-bit variant. Related key attacks can clearly be avoided by always selecting cryptographic keys independently at random.

A bi-clique technique can be applied to the cipher to reduce the complexity of exhaustive key search. For example in [53] it is shown that one can break AES-128 with $2^{126.2}$ encryption operations and 2^{88} chosen plaintexts. For AES-192 and AES-256 these numbers become $2^{189.7}/2^{40}$ and $2^{254.4}/2^{80}$ respectively.

Camellia

The Camellia block cipher is used as one of the possible cipher suites in TLS, and unlike AES is of a Feistel cipher design. Camellia has a block length of 128 bits and supports 3 key lengths: 128, 192 and 256 bits [228]. The versions with a 192- or a 256-bit key are 33% slower than the versions with a 128-bit key.

Observation: Just as for AES there is a relatively simple set of algebraic equations which define the Camellia transform; this might leave it open to algebraic attacks. However, just like AES such attacks have not been shown to be effective.

3.2.2 Legacy Block Ciphers

3DES

Comes in two variants; a two key version with a 112-bit key and a three key version with a 168-bit key [266]. The effective key length of three key 3DES is 112 bits and not 168 bits as one would expect. The small block length (64-bits) is a problem in some applications. The two key variant suffers from a (theoretical) related key attack, however if the two keys are selected uniformly at random this is not a problem in practice.

Observation: Due to the iterated nature of the cipher the security is not as strong as the key length would suggest. For the two key variant the security is 2^{120-t} where 2^t plaintext/ciphertext pairs are obtained; for the three key variant the security is reduced to 2^{112}.

Kasumi

This cipher [118], used in 3GPP, has a 128-bit key and 64-bit block size is a variant of MIST-1. Kasumi is called UIA1 in UMTS and is called A5/3 in GSM

Observation: Whilst some provable security against linear and differential cryptanalysis has been established [188], the cipher suffers from a number of problems. A related key attack [41] requiring 2^{76} operations and 2^{54} plaintext/ciphertext pairs has been presented. In [98] a more efficient related key attack is given which requires 2^{32} time and 2^{26} plaintext/ciphertext pairs. These attacks *do not affect* the practical use of Kasumi in applications such as 3GPP, however given them we do not recommend Kasumi for use in further applications.

Blowfish

This cipher [312] has a 64-bit block size, which is too small for some applications and the reason we only recommend it for legacy use. It also has a key size ranging from 32- to 448-bits, which we clearly only recommend using at 80-bits and above for legacy applications. The Blowfish block cipher and is used in some IPsec configurations.

OBSERVATION: There have been a number of attacks on reduced round versions [189, 293, 334] but no attacks on the full cipher.

3.2.3 Historical (not-recommended) Block Ciphers

DES

DES has a 56-bit key and 64-bit block size and so is not considered secure by today's standards. It is susceptible to linear [42] and differential cryptanalysis [229].

3.3 Hash Functions

Hash function outputs should be, in our opinion, a minimum of 160 bits in length for legacy applications and 256 bits in length for all new applications. Hash functions are probably the area of cryptography which has had the most attention in the past decade. This is due to the spectacular improvements in the cryptanalysis of hash functions, as well as the subsequent SHA-3 competition to design a replacement for our existing set of functions. Most existing hash functions are in the Merkle–Damgård family, and derive much of their design philosophy from the MD-4 hash function; such hash fucntions are said to be in the MD-X family. This family includes MD-4, MD-5, RIPEMD-128, RIPEMD-160, SHA-1 and SHA-2.

	Output	Recommendation	
Primitive	Lenfth	Legacy	Future
SHA-2	256, 384, 512	✓	✓
SHA-3	?	✓	✓
Whirlpool	512	✓	✓
SHA-2	224	✓	✗
RIPEMD-160	160	✓	✗
SHA-1	160	✓	✗
MD-5	128	✗	✗
RIPEMD-128	128	✗	✗

Table 3.3: Hash Function Summary

3.3.1 Recommended Hash Functions

SHA-2

SHA-2 is actually a family of four algorithms, SHA-224, SHA-256, SHA-384 and SHA-512. SHA-224 (resp. SHA-384) is a variant itself of SHA-256 (resp. SHA-512), but just uses a different IV and then truncates the output. Due to our decision of symmetric security lengths of less than 128 being only suitable for legacy applications we denote SHA-224 as in the legacy only division of our analysis.

Observation: For SHA-224/SHA-256 (resp. SHA-384/SHA-512) reduced round collision attacks 31 out of 64 (resp. 24 out of 80) have been reported [162, 239, 306]. In addition reduced round variants 43 (resp. 46) have also been attacked for preimage resistance [22, 138].

SHA-3

The competition organized by NIST to find an algorithm for SHA-3 ended on October 2nd, 2012, with the selection of Keccak [131]. Currently, NIST is drafting a FIPS describing SHA-3.

Observation: Reduced round collision attacks (8 out of 24) have been reported [96].

Whirlpool

Whirlpool produces a 512-bit hash output and is not in the MD-X family; being built from AES style methods, thus it is a good alternative to use to ensure algorithm diversity.

Observation: Preimage attacks on 5 (out of 10) rounds have been given [307], as well as collisions on 5.5 rounds [210], with complexity 2^{120}.

3.3.2 Legacy Hash Functions

RIPEMD-160

It is anticipated that collision attacks on RIPEMD-160 are likely to be found on reduced round versions in the near future [240]. Collision attacks on 36 rounds (out of 80) have already been found [238].

SHA-1

SHA-1 is in widespread use and was designed to provide protection against collision finding of 2^{80}. However, the analysis of MD-5 has been extended to SHA-1 (which is in the same family), resulting in collisions being found in 2^{69} operations [341], and even lower complexity [241, 340]. The current best analysis is that of $2^{57.5}$ operations, reported in [324]. On the other hand explicit collisions for the full SHA-1 have not yet been found, despite collisions for a reduced round variant (73 rounds out of 80) being found [102]. An extension of Stevens' analysis from [324] is expected to yield a full collision in 2^{61} operations.

<u>Observation:</u> The literature also contains preimage attacks on a variant reduced to 45-48 rounds [23, 69].

3.3.3 Historical (not recommended) Hash Functions

MD-5

Despite being widely deployed the MD-5 hash function should not be considered secure. Collisions can be found within seconds on a modern desktop computer. The literature on the collision weakness of MD-5 and its impact in various scenarios is wide [219, 309, 325–327]. Preimage resistance can also be broken in time $2^{124.4}$ [308].

RIPEMD-128

Given an output size of 128-bits, collisions can be found in RIPEMD-128 in time 2^{64} using generic attacks, thus RIPEMD-128 can no longer be considered secure in a modern environment irrespective of any cryptanalysis which reduces the overall complexity. Practical collisions for a 3-round variant were reported in 2006, [240]. In [211] futher cryptanalytic results were presented which lead one to conclude that RIPEMD-128 is not to be considered secure.

3.4 Stream Ciphers

Generally speaking stream ciphers should be used with a distinct IV for each message, unless the key is used in a one-time manner (as for example in a DEM construction). Again, for each cipher we give a short description of state of the art with respect to known attacks, we then give recommendations for minimum parameter sizes for future and legacy use. For convenience these recommendations are summarized in Table 3.4. Where possible, it is probably better to use a block cipher in mode such as CTR mode than a dedicated stream cipher. Dedicated stream ciphers offer performance advantages over AES in CTR mode, but historically the science of stream cipher design lags that of block cipher and mode of operation design.

3.4.1 Recommended Stream Ciphers

Rabbit

Rabbit was an entrant to the eSTREAM competition and included in the final eSTREAM portfolio as promising for software implementations. Rabbit uses a 128-bit key together with a 64-bit IV. Rabbit is described in RFC 4503 and is included in ISO/IEC 18033-4 [167].

	Recommendation	
Primitive	Legacy	Future
Rabbit	✓	✓
SNOW 3G	✓	✓
Trivium	✓	✗
SNOW 2.0	✓	✗
A5/1	✗	✗
A5/2	✗	✗
E0	✗	✗
RC4	✗	✗

Table 3.4: Stream Cipher Summary

SNOW 3G

SNOW 3G is an enhanced version of SNOW 2.0, the main change being the addition of a second S-Box as a protection against future advances in algebraic cryptanalysis. It uses a 128-bit key and a 128-bit IV. The cipher is the core of the algorithms UEA2 and UIA2 of the 3GPP UMTS system, which are identical to the algorithms 128-EIA1 and 128-EEA1 in LTE.

3.4.2 Legacy Stream Ciphers

Trivium

Trivium was an entrant to the eSTREAM competition and included in the final eSTREAM portfolio as promising for hardware implementations. It has been included in ISO/IEC 29192-3 on lightweight stream ciphers [169]. Trivium uses an 80-bit key together with an 80-bit IV.

Observation: There has been a number of papers on the cryptanalysis of Trivium and there currently exists no attack against full Trivium. Aumasson et al. [25] present a distinguishing attack with complexity 2^{30} on a variant of Trivium with the initialization phase reduced to 790 rounds (out of 1152). Maximov and Biryukov [232] present a state recovery attack with time complexity around $2^{83.5}$. This attack shows that Trivium with keys longer than 80 bits provides no more security than Trivium with an 80-bit key. It is an open problem to modify Trivium so as to obtain 128-bit security in the light of this attack.

SNOW 2.0

SNOW 2.0 comes in a 128 and 256-bit key variants. A distinguishing attack against SNOW 2.0 is theoretically possible [268], but it requires 2^{174} bits of key-stream and work. Given the introduction of SNOW 3G we only see a legacy application for SNOW 2.0, and recommend all new systems wishing to use a SNOW-like cipher to use SNOW 3G.

3.4.3 Historical (non recommended) Stream Ciphers

A5/1

A5/1 was originally designed for use in the GSM protocol. It is initialized using a 64-bit key and a publicly known 22-bit frame number. The design of A5/1 was initially kept secret until 1994 when the general design was leaked and has since been fully reverse engineered. The cipher has been subject to a number of attacks. The best attack was shown to allow for real-time decryption of GSM mobile phone conversations [27]. As result this cipher is *not* considered to be secure.

A5/2

A5/2 is a weakened version of A5/1 to allow for (historic) export restrictions to certain countries. It is therefore *not* considered to be secure.

E0

The E0 stream cipher is used to encrypt data in Bluetooth systems. It uses a 128-bit key and no IV. The best attack recovers the key using the first 24 bits of 2^{24} frames and 2^{38} computations [223]. This cipher is therefore *not* considered to be secure.

RC4

RC4 comes in various key sizes. Despite widespread deployment the RC4 cipher has for many years been known to suffer from a number of weaknesses. There are various distinguishing attacks [226], and state recovery attacks [233]. (An efficient technique to recover the secret key from an internal state is described in [40].)

An important shortcoming of RC4 is that it was designed without an IV input. Some applications, notably WEP and WPA "fix" this by declaring some bytes of the key as IV, thereby effectively enabling related-key attacks. This has led to key-recovery attacks on RC4 in WEP [337]. When initialized the first 512 output bytes of the cipher should be discarded due to statistical biases. If this step is omitted, then key-recovery attacks can be accelerated, e.g. those on WEP and WPA [317].

Despite statistical biases being known since 1995, SSL/TLS does not discard any of the output bytes of RC4; this results in a recent attacks by AlFardan et al. [12] and Isobe et al. [163].

3.5 Public Key Primitives

For each primitive we give a short description of state of the art with respect to known attacks, we then give recommendations for minimum parameter sizes for future and legacy use. For convenience these recommendations are summarized in Table 3.5. In the table we let $\ell(\cdot)$ to denote the logarithm to base two of a number; a $\star$ denotes some conditions which also need to be tested which are explained in the text.

Primitive	Parameters	Legacy System Minimum	Future System Minimum
RSA Problem	N, e, d	$\ell(n) \geq 1024$, $e \geq 3$ or 65537, $d \geq N^{1/2}$	$\ell(n) \geq 3072$ $e \geq 3$ or 65537, $d \geq N^{1/2}$
Finite Field DLP	p, q, n	$\ell(p^n) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^n) \geq 3072$ $\ell(p), \ell(q) > 256$
ECDLP	p, q, n	$\ell(q) \geq 160$, $\star$	$\ell(q) > 256$, $\star$
Pairing	p, q, n, d, k	$\ell(p^{k \cdot n}) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^{k \cdot n}) \geq 3072$ $\ell(p), \ell(q) > 256$

Table 3.5: Public Key Summary

3.5.1 Factoring

Factoring is the underlying hard problem behind all *schemes* in the RSA family. In this section we discuss what is known about the *mathematical* problem of factoring, we then specialize to the *mathematical* understanding of the RSA Problem. The RSA Problem is the underlying cryptographic primitive, we are not considering the RSA encryption or signature algorithm at this point. In fact vanilla RSA should *never* be used as an encryption or signature algorithm, the RSA primitive (i.e. the RSA Problem) should only be used in combination with one of the well defined schemes from Chapter 4.

Since the mid-1990s the state of the art in factoring numbers of general form has been determined by the factorization of the RSA-challenge numbers. In the last decade this has progressed at the following rate RSA-576 (2003) [124], RSA-640 (2005) [125], RSA-768 (2009) [202]. These records have all been set with the Number Field Sieve algorithm [216]. It would seem prudent that only legacy applications should use 1024 bit RSA modulus going forward, and that future systems should use RSA keys with a minimum size of 3072 bits.

Since composite moduli for cryptography are usually chosen to be the product of two large primes $N = p \cdot q$, to ensure they are hard to factor it is important that p and q are chosen of the same bit-length, but not too close together. In particular

- If $\ell(p) \ll \ell(q)$ then factoring can be made easier by using the small value of p (via the ECM method [186]). Thus selecting p and q such that $0.1 < |\ell(p) - \ell(q)| \leq 20$, is a good choice.

- On the other hand if $|p - q|$ is less than $N^{1/4}$ then factoring can be accomplished by the Coppersmith's method [79].

Selecting p and q to be random primes of bit length $\ell(N)/2$ will, with overwhelming probably, ensure that N is hard to factor with both these techniques.

RSA Problem

Cryptosystems based on factoring are actually usually based not on the difficulty of factoring but on the difficulty of solving the RSA problem. The RSA Problem is defined to be that of given an RSA modulus $N = p \cdot q$, an integer value e such that $\gcd(e, (p-1) \cdot (q-1)) = 1$, and a value $y \in \mathbb{Z}/N\mathbb{Z}$ find the value $x \in \mathbb{Z}/N\mathbb{Z}$ such that $x^e = y \pmod{N}$.

If e is too small such a problem can be easily solved, assuming some side information, using Coppersmith's lattice based techniques [77, 78, 80]. Thus for RSA based encryption schemes it is common to select $e \geq 65537$. For RSA based signature schemes such low values of e do not seem to be a problem, thus it is common to select $e \geq 3$. For efficiency one often takes e to be as small a prime as the above results would imply; thus it is very common to find choices of $e = 65537$ for encryption and $e = 3$ for signatures in use.

The RSA private key is given by $d = 1/e \pmod{(p-1) \cdot (q-1)}$. Some implementers may be tempted to choose d "small" and then select e so as to optimize the private key operations. Clearly, just from naive analysis d cannot be too small. However, lattice attacks can also be applied to choices of d less than $N^{0.292}$ [57, 343]. Lattice attacks in this area have also looked at situations in which some of the secret key leaks in some way, see for example [116, 150]. We therefore recommend that d is chosen such that $d > N^{1/2}$, this will happen with overwhelming probability if the user selects e first and then finds d.

3.5.2 Discrete Logarithms

The discrete logarithm problem can be defined in any finite abelian group. The basic construction is to take a finite abelian group of large prime order q generated by an element g. The discrete logarithm problem is to recover $x \in \mathbb{Z}/q\mathbb{Z}$ from the value $h = g^x$. It is common for the group and generator to be used by a set of users; in this case the tuple $\{\langle g \rangle, q\}$ is called a set of *Domain Parameters*.

Whilst the DLP is the underlying number theoretic problem in schemes based on the discrete logarithm problem, actual cryptographic schemes base their security on (usually) one of three related problems; this is similar to how factoring based schemes are usually based on the RSA problem and not factoring per se. The three related problems are:

- Computational Diffie–Hellman problem: Given g^x and g^y for hidden x and y compute $g^{x \cdot y}$.
- Decision Diffie–Hellman problem: Given g^x, g^y and g^z for hidden x, y and z decide if $z = x \cdot y$.

- Gap Diffie–Hellman problem: Given g^x and g^y for hidden x and y compute $g^{x \cdot y}$, given an oracle which allows solution of the Decision Diffie–Hellman problem.

Clearly the ability to solve the DLP will also give one the ability to solve the above three problems, but the converse is not known to hold in general (although it is in many systems widely believed to be the case).

Finite Field DLP

The discrete logarithm problem in finite fields (which we shall refer to simply as DLP), and hence the Diffie–Hellman problem, Decision Diffie–Hellman problem and gap Diffie–Hellman problem, is parametrized by the finite field $\mathbb{F}_{p^n}$ and the subgroup size q, which should be prime. In particular this means that q divides $p^n - 1$. To avoid "generic attacks" the value q should be at least 160 bits in length for legacy applications and at least 256 bits in length for new deployments.

For the case of small prime characteristic, i.e. $p = 2, 3$ there is new algorithm was presented in early 2013 by Joux [184] which runs in time $L(1/4 + o(1))$, for when the extension degree n is composite (which are of relevance to pairing based cryptography). This algorithm was quickly supplanted by an algorithm which runs in quasi-polynomial time by Barbulescu and others [26]. Also in 2013 a series of record breaking calculations were performed by a French team and a Irish team for characteristic two fields, resulting in the records of $\mathbb{F}_{2^{6120}}$ [137] and $\mathbb{F}_{2^{6168}}$ [182]. For characteristic three the record is $\mathbb{F}_{3^{582}}$ [332]. For prime values of n the best result is a discrete logarithm calculation in the field $\mathbb{F}_{2^{809}}$ [61]. All of these results make use of special modification to the function field sieve algorithm [9]. In light of these results no system should be deployed relying on the hardness of the DLP in small characteristic fields.

For large prime fields, i.e. $n = 1$, the algorithm of choice is a variant of the Number Field Sieve [135]. The record here is for a finite field $\mathbb{F}_p$ with p a 530 bit prime [201] set in 2007. In light of the "equivalence" between the number field sieve for factoring and that for discrete logarithms our recommendation is in this case that legacy applications should use 1024 bit p, and new systems should use a minimum p of 3072 bits.

There has been some work on the case of so called medium prime fields; fields with p larger than 100 and $1 < n < 100$, see for example [183,185]. Currently these algorithms have no cryptographic impact; although this might change if the fields being considered have impact on pairing based cryptography (see Section 3.5.3). This is because all pairing based applications have $\log_2(p) \geq 160$.

ECDLP

Standard elliptic curve cryptography (i.e. ECC not using pairings) comes in two flavors, either systems are based on elliptic curves over a large prime field $E(\mathbb{F}_p)$, or they are based on elliptic curves over a field of characteristic two $E(\mathbb{F}_{2^n})$. We denote the field size by p^n in what follows, so when writing p^n we implicitly assume either $p = 2$ or $n = 1$. We let q denote the largest prime factor of the group order and let h denote the "cofactor", so $h \cdot q = \#E(\mathbb{F}_{p^n})$. To avoid known attacks one selects these parameters so that

- The smallest t such that q divides $p^{t \cdot n} - 1$ is such that extracting discrete logarithms in the finite field of size $p^{t \cdot n}$ is hard. This is the so called MOV condition [242].
- If $n = 1$ then we should not have $p = q$. These are the so-called anomalous curves for which there is a polynomial time attack [310, 316, 321].
- If $p = 2$ then n should be prime. This is to avoid so-called Weil descent attacks [132].

It is common, to avoid small subgroup attacks, for the curve to be chosen such that $h = 1$ in the case of $n = 1$ and $h = 2$ or 4 in the case of $p = 2$. There are a subclass of curves called *Koblitz curves* in the case of $p = 2$ which offer some performance advantages, but we do not consider the benefit to outweigh the cost for modern processors thus our discussion focuses on general curves only. Some standards, e.g. [117] stipulate that the class number of the associated endomorphism ring must be larger than some constant (e.g. 200). We see no cryptographic reason for making this recommendation; since by choosing random curves it is over whelmingly likely to occur in any case and no weakness is known for such curves.

The largest ECDLP records have been set for the case of $n = 1$ with a p of size 109-bits [60], and for $p = 2$ with $n = 109$ [72]. These record setting achievements are all performed with the method of distinguished points [333], which is itself based on Pollard's rho method [285]. To avoid such "generic attacks" the value q should be at least 160 bits in length for legacy applications and at least 256 bits in length for new deployments.

Various standards, e.g. [19, 20, 315] specify a set of recommended curves; many of which also occur in other standards and specifications, e.g. in TLS [50]. Due to issues of interoperability the authors feel that using a curve specified in a standard is best practice. Thus the main choice for an implementer is between curves in characteristic two and large prime characteristic.

3.5.3 Pairings

Pairing based systems take two elliptic curves $E(\mathbb{F}_{p^n})$ and $\hat{E}(\mathbb{F}_{p^{n \cdot d}})$, each containing a subgroup of order q. We denote the subgroup of order q in each of these elliptic curves by $\mathbb{G}_1$ and $\mathbb{G}_2$. Pairing based systems also utilize a finite field $\mathbb{F}_{p^{k \cdot n}}$, where q divides $p^{k \cdot n} - 1$. These three structures are linked via a bilinear mapping $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, where G_T is the multiplicative subgroup of $\mathbb{F}_{p^{k \cdot n}}$ of order q. The value k is called the embedding degree, and we always have $1 \leq d \leq k$. Whilst there are many hard problems on which pairing based cryptography is based, the most efficient attack is almost always the extraction of discrete logarithms in either one of the elliptic curves or the finite field (although care needs to be taken with some schemes due to the additional information the scheme makes available). Given our previous discussion on the finite field DLP and the ECDLP the parameter choices for legacy and new systems are immediate. Note: This immediately implies that $\log_2(p) > 160$, so as to avoid attacks based on the low characteristic discrete logarithm algorithms in finite fields. In addition, note that the conditions in Table 3.5 immediately imply all the special conditions for elliptic curve based systems indicated by a $\star$ in the ECDLP row.

3.6 Recommendations

Recommending key sizes for long term use is somewhat of a hit-and-miss affair, for a start it assumes that the algorithm you are selecting a key size for is not broken in the mean time. So in recommending key sizes for specific application domains we make an *implicit* assumption that the primitive, scheme or protocol which utilizes this key size is not broken in the near future. All primitives, protocols and schemes marked as suitable for future use in this document we have confidence will remain secure for a significant period of time.

Making this assumption still implies a degree of choice as to key size however. The AES block cipher may remain secure for the next fifty years, but one is likely to want to use a larger key size for data which one wishes to secure for fifty years as opposed to, say, five years. Thus in recommending key sizes we make two distinct cases for schemes relevant for future use. The first cases is for security which you want to ensure for *at least* ten years (which we call *near term*), and secondly for security for thirty to fifty years (which we call *long term*). Again we reiterate these are purely key size recommendations and they do not guarantee security, nor do they guarantee against attacks on the underlying mathematical primitives.

In Table 3.6 we present our explicit key size recommendations. The reader will see that we have essentially followed the NIST equivalence [265] between the different key sizes. However, these key sizes equivalences need to be understood to apply only to the "best in class" algorithm for block ciphers, hash function, RSA parameters, etc etc. It is clearly possible for a block cipher of 128-bits security to not offer 128-bit security due to cryptanalytic attacks.

We have focused on 128 bit security in this document for future use recommendations; clearly this offers a good long term security gaurantee. It is plausible that a similar recommendation could be made at (say) the 112 bit security level (which would correspond to roughly 2048 bit RSA keys). The line has to be drawn somewhere and there is general agreement this should be above the 100-bit level; whether one selects 112 bits or 128 bits as the correct level is a matter of taste. Due to the need to protect long term data we have taken the conservative choice and settled on 128 bits; with a higher level for very long term use.

Due to the problem of key sizes not being a good measure of security on their own, and also due to considerations of underlying performance costs, at the time of writing the recommendation for future use can be summarized in the following simple choices:

1. Block Ciphers: For near term use we recommend AES-128 and for long term use AES-256.
2. Hash Functions: For near term use we recommend SHA-256 and for long term use SHA-512.
3. Public Key Primitive: For near term use we recommend 256 bit elliptic curves, and for long term use 512 bit elliptic curves.

	Parameter	Legacy	Future System Use Near Term	Future System Use Long Term
Symmetric Key Size	k	80	128	256
Hash Function Output Size	m	160	256	512
MAC Output Size	m	80	128	$256^\star$
RSA Problem	$\ell(n) \geq$	1024	3072	15360
Finite Field DLP	$\ell(p^n) \geq$	1024	3072	15360
	$\ell(p), \ell(q) \geq$	160	256	512
ECDLP	$\ell(q) \geq$	160	256	512
Pairing	$\ell(p^{k \cdot n}) \geq$	1024	3072	15360
	$\ell(p), \ell(q) \geq$	160	256	512

Table 3.6: Key Size Recommendations. A $\star$ notes the value could be smaller due to specific protocol or system reasons, the value given is for general purposes.

Finally, we note that the recommendations above, and indeed all analysis in this document, is on the basis that there is no breakthrough in the construction of quantum computers.

Chapter 4

Schemes

As mentioned previous a cryptographic scheme usually comes with an associated security proof. This is (most often) an algorithm which takes an adversary against the scheme in some well defined model, and turns the adversary into one which breaks some property of the underlying primitive (or primitives) out of which the scheme is constructed. If one then believes the primitive to be secure, one then has a strong guarantee that the scheme is well designed. Of course other weaknesses may exist, but the security proof validates the basic design of the scheme. In modern cryptography all schemes should come with a security proof.

The above clean explanation however comes with some caveats. In theoretical cryptography a big distinction is made between schemes which have proofs in the *standard model* of computation, and those which have proofs in the *random oracle model*. The random oracle model is a model in which hash functions are assumed to be idealized objects. A similar issue occurs with some proofs using idealized groups (the so-called *generic group model*), or idealized ciphers (a.k.a the *ideal cipher model*). In this document we take, as do most cryptographers working with real world systems, the pragmatic view; that a scheme with a proof in the random oracle model is better than one with no proof, and that the use of random oracles etc can be justified if they produce schemes which have performance advantages over schemes which have proofs in the standard model.

It is sometimes tempting for an implementer to use the same key for different purposes. For example to use a symmetric AES key as both the key to an application of AES in an encryption scheme, and also for the use of AES within a MAC scheme, or within different modes of operation [134]. As another example one can imagine using an RSA private key as both an decryption key and as a key to generate RSA signatures; indeed this latter use-case is permitted in the EMV chip-and-pin system [90]. Another example would be to use the same encryption key on a symmetric channel between Alice and Bob for two way communication, i.e. using has one bidirectional key as opposed to two unidirectional keys. Such usage can often lead to unexpected system behaviour, thus it is good security practice to design into systems explicit *key separation*.

Key separation means we can isolate the systems dependence on each key and its usages; and indeed many security proofs implicitly assume that key separation is being deployed. However, in some *specific* instances one can show, for specific pairs of cryptographic schemes, that key separation is not necessary. We do not discuss this further in this document but refer the reader to [15,90,274], and simply warn the reader to violate the key separation principle with caution.

In Tables 4.1, 4.2, 4.3 and 4.4 we present our summary of the various symmetric and asymmetric schemes considered in this document. In each scheme we assume the parameters and building blocks have been chosen so that the recommendations of Chapter 3 apply.

In 4.1 we give (some of) the security notions for symmetric encryption achieved by the the various constructions presented in Sections 4.1 and 4.3. Whether it is suitable for future or legacy use needs to be decided by consideration of the underlying block cipher and therefore by reference to Table 3.2. For general encryption of data we *strongly* recommend the use of an authenticated encryption scheme, and CCM, EAX or GCM modes in particular. The columns IND-CPA, IND-CCA and IND-CVA refer to indistinguishablity under chosen plaintext, chosen ciphertext and ciphertext validity attacks. The latter class of attacks lie somewhere between IND-CPA and IND-CCA and include padding oracle attacks. Of course some of the padding oracle attacks imply a specific choice as to how padding is performed in such schemes. In our table a scheme which does not meet IND-CVA does not meet IND-CVA for a *specific* padding method. Similarly an authenticated encryption scheme which does not meet IND-CCA is one which does not meet this goal for a *specific* choice of underlying components.

4.1 Block Cipher Basic Modes of Operation

In this section we detail the main modes of operation for using a block cipher as a symmetric encryption scheme. Note, we leave a discussion of schemes which are secure against chosen-ciphertext attacks until Section 4.3; thus this section is essentially about IND-CPA schemes only. As such *all* schemes in this section need to be used with extreme care in an application. Further technical discussion and comparison on the majority of modes stated here can be found in [297].

4.1.1 ECB

Electronic Code Book (ECB) mode [258] should be used with care. It should only be used to encrypt messages with length at most that of the underlying block size, and only for keys which are used in a one-time manner. This is because without such guarantees ECB mode provides no modern notion of security.

4.1.2 CBC

Cipher Block Chaining (CBC) mode [258] is the most widely used mode of operation. Unless used with a one-time key, an independent and random IV *must* be used for each message; with such a usage the mode can be shown to be IND-CPA secure [30], if the underlying block cipher is secure.

Scheme	IND-CPA	IND-CVA	IND-CCA	Notes
		Block Cipher Modes of Operation		
OFB	✓	(✓)	✗	No padding
CFB	✓	(✓)	✗	No padding
CTR	✓	(✓)	✗	No padding
CBC	✓	✗	✗	
ECB	✗	✗	✗	See text
XTS	-	-	✗	See text
EME	-	-	✗	See text
		Authenticated Encryption		
Encrypt-then-MAC	✓	✓	✓	Assuming secure Encrypt/MAC used
OCB	✓	✓	✓	
CCM	✓	✓	✓	Superseded by EAX
EAX	✓	✓	✓	
CWC	✓	✓	✓	
GCM	✓	✓	✓	
MAC-then-Encrypt	✓	✗	✗	See Encrypt-then-MAC text
Encrypt-and-MAC	✓	✗	✗	See Encrypt-then-MAC text

Table 4.1: Symmetric Key Encryption Summary Table

	Recommendation		
Scheme	Legacy	Future	Building Block
EMAC	✓	✓	Any block cipher as a PRP
CMAC	✓	✓	Any block cipher as a PRP
HMAC	✓	✓	Any hash function as a PRF
UMAC	✓	✓	An internal universal hash function
GMAC	✓	✗	Finite field operations
AMAC	✓	✗	Any block cipher

Table 4.2: Symmetric Key Based Authentication Summary Table

The mode is not IND-CCA secure as ciphertext integrity is not ensured, for applications requiring IND-CCA security an authenticated encryption mode is to be used (for example by applying a message authentication code to the output of CBC encryption). For further details see Section 4.3.

Since CBC mode requires padding of the underlying message before encryption the mode suffers from certain padding oracle attacks [276, 335, 346]. Again usage of CBC within an authenticated encryption scheme (and uniform error reporting) can mitigate against such attacks.

Primitive	Recommendation Legacy	Recommendation Future	Building Block
NIST-800-108-KDF(all modes)	✓	✓	A PRF (e.g. a MAC)
X9.63-KDF	✓	✓	Any hash function
NIST-800-56-KDF-A/B	✓	✓	Any hash function
NIST-800-56-KDF-C	✓	✓	A MAC function
IKE-v2-KDF	✓	✓	HMAC used as a PRF
TLS-v1.2-KDF	✓	✓	HMAC (SHA-2) as a PRF
IKE-v1-KDF	✓	✗	HMAC used as a PRF
TLS-v1.1-KDF	✓	✗	HMAC (MD-5 and SHA-1) used as a PRF

Table 4.3: Key Derivation Function Summary Table

4.1.3 OFB

Output Feedback (OFB) mode [258] produces a stream cipher from a block cipher primitive, using an IV as the initial input to the block cipher and then feeding the resulting output back into the blockcipher to create a stream of blocks. To improve efficiency the stream can be precomputed.

The mode is IND-CPA secure when the IV is random (this follows from the security result for CBC mode). If the IV is a nonce then IND-CPA security is not satisfied. The mode is not IND-CCA secure as ciphertext integrity is not ensured, for applications requiring IND-CCA security an authenticated encryption mode is to be used (cf. Section 4.3). OFB mode does not require padding so does not suffer from padding oracle attacks.

4.1.4 CFB

Cipher Feedback (CFB) mode [258] produces a self-synchronising stream cipher from a block cipher. Unless used with a one-time key the use of an independent and random IV*must* be used for each message; with such a usage the mode can be shown to be IND-CPA secure [14], if the underlying block cipher is secure.

The mode is not IND-CCA secure as ciphertext integrity is not ensured. For applications requiring IND-CCA security an authenticated encryption mode is to be used (cf. Section 4.3). CFB mode does not require padding so does not suffer from padding oracle attacks.

4.1.5 CTR

Counter (CTR) mode [258] produces a stream cipher from a block cipher primitive, using a counter as the input message to the block cipher and then taking the resulting output as the stream cipher sequence. The counter (or IV) should be a nonce to achieve IND-CPA security [30]. The scheme is rendered insecure if the counter is repeated.

Scheme	Recommendation Legacy	Future	Notes
Public Key Encryption/Key Encapsulation			
RSA-OAEP	✓	✓	See text
RSA-KEM	✓	✓	See text
PSEC-KEM	✓	✓	See text
ECIES-KEM	✓	✓	See text
RSA-PKCS# 1 v1.5	✗	✗	Should only be used in special situations
Public Key Signature Schemes			
RSA-PSS	✓	✓	See text
ISO-9796-2 RSA-DS2	✓	✓	Message recovery variant of RSA-PSS
PV Signatures	✓	✓	ISO 14888-3 only defines these for a finite field
(EC)Schnorr	✓	✓	See text
RSA-PKCS# 1 v1.5	✓	✗	No security proof
RSA-FDH	✓	✗	Issues in instantiating the required hash function
ISO-9796-2 RSA-DS3	✓	✗	Similar to RSA-FDH
(EC)DSA,(EC)GDSA	✓	✗	Weak provable security guarantees
(EC)KDSA,(EC)RDSA	✓	✗	Weak provable security guarantees
ISO-9796-2 RSA-DS1	✗	✗	Attack exists (see notes)
Identity Based Encryption			
BB	✓	✓	See text
SK	✓	✓	See text
BF	✓	✗	See text

Table 4.4: Public Key Based Scheme Summary Table

The mode is not IND-CCA secure as ciphertext integrity is not ensured, for applications requiring IND-CCA security an authenticated encryption mode is to be used (cf. Section 4.3). No padding is necessary so the mode does not suffer from padding oracle attacks.

Unlike all previous modes mention, CTR mode is easily and fully parallelisable allowing for much faster encryption and decryption.

We recommend this mode above all others when privacy-only encryption is required.

4.1.6 XTS

XTS mode [261] is short for *XEX Tweakable Block Cipher with Ciphertext Stealing* and is based on the XEX tweakable block cipher [296] (using two keys instead of one). The mode was specifically designed for encrypted data storage using fixed-length data units, and is used in the TrueCrypt system.

Due to the specific application of disc encryption the standard notion of IND-CPA security is not appropriate for this setting. It is mentioned in [261] that the mode should provide slightly more protection against data manipulation than standard confidentiality-only modes. The exact notion remains unclear and as a result XTS mode does not have a proof of security. Further technical discussion on this matter can be found in [297, Chapter 6] and [222]. The underlying tweakable block cipher XEX is proved secure as a strong pseudorandom permutation [296].

Due to its "narrow-block" design XTS mode offers significant efficiency benefits over "wide-block" schemes.

4.1.7 EME

ECB-mask-ECB (EME) mode was designed by Halevi and Rogaway [141] and has been improved further by Halevi [139]. EME mode is design for the encrypted data storage setting and is proved secure as a strong tweakable pseudorandom permutation. Due to its wide block design it will be half the speed of XTS mode but in return does offer greater security. EME is patented and its use is therefore restricted.

4.2 Message Authentication Codes

Message Authentication Codes (MAC) are symmetric-key cryptosystems that aim to achieve message integrity. Most commonly used designs fall in one of two categories: block-cipher based schemes (detailed in Section 4.2.1), and hash function based schemes (Sections 4.2.3).

4.2.1 Block Cipher Based MACs

Almost all block cipher based MACs are based on CBC-MAC. The essential differences in application arise due to the padding method employed, how the final iteration is performed and the post-processing method needed to produce the final output. The final iteration and post-processing methods impact on the number of keys required by the MAC function. The ISO 9797-1 standard [171] defines four padding methods, three final iteration methods and three post-processing methods, and from these it defines six CBC-MAC algorithms which can be utilized with any cipher; one of which uses a non-standard processing of the first block. In summary of these six algorithms we have, where H_q is the output of the final iteration, H_{q-1} is the output of the penultimate iteration, D_i is the i padded message block, and K is the block cipher key used for iterations $1, \ldots, q-1$,

ISO 9797-1 Number	First Iteration	Final Iteration	Post Processing	a.k.a
1	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = H_q$	CBC-MAC
2	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_{K'}(H_q)$	EMAC
3	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_K(D_{K'}(H_q))$	AMAC
4	$H_1 = E_{K''}(E_K(D_1))$	$H_q = E_K(D_q \oplus H_{q-1})$	$G = E_{K'}(H_q)$	-
5	$H_1 = E_K(D_1)$	$H_q = E_K(D_q \oplus H_{q-1} \oplus K')$	$G = H_q$	CMAC
6	$H_1 = E_K(D_1)$	$H_q = E_{K'}(D_q \oplus H_{q-1})$	$G = H_q$	LMAC

Of these we only focus on EMAC, AMAC and CMAC, being the most utilized, of these we do not recommend AMAC for other than legacy use (since EMAC and CMAC are easily used instead, and offer stronger guarantees), and vanilla CBC-MAC is on its own not considered secure. Of course all MACs should be used with keysizes and output sizes which match our key size recommendations. In particular this means for high security levels, i.e. equivalent to 256-bits of AES security, these MAC functions cannot be used with AES as there output lengths are limited to the block length of the underlying block cipher. Unless the application allows shorter MAC values for some reason.

EMAC

The Algorithm was introduced in [279] and is specified as Algorithm 2 in ISO-9797-1 [171]. Provable security guarantees have been derived in [279, 280]. Note that the guarantees are for the version of the scheme that uses two independent keys; there are no known guarantees for the version where the two keys are derived from a single key in the way specified by the standard. The function LMAC obtains the same security bounds as EMAC but uses one fewer encryption operation.

AMAC

The algorithm was introduced in [16] and is also specified as Algorithm 3 in ISO 9797-1 [171]. The algorithm is known as ANSI Retail MAC, or just AMAC for short, and is deployed in banking applications with DES as the underlying block cipher. There are known attacks against the scheme that require $2^{n/2}$ MAC operations, where n is the block size. The scheme should therefore not be used, unless frequent rekeying is employed.

CMAC

The CMAC scheme was introduced in [173] and standardized as Algorithm 5 in [171]. It enjoys provable security guarantees under the assumption that the underlying block-cipher is a PRP [250]. In particular this requires frequent rekeying; for example when instantiated with AES-128 existing standards recommend that the scheme should be used for at most 2^{48} messages. Furthermore, the scheme should only be used in applications where no party learns the enciphering of the all-0 string under the block-cipher underlying the MAC scheme.

4.2.2 GMAC

GMAC is the MAC function underlying the authenticated encryption mode GCM. It makes use of polynomials over the finite field $GF(2^{128})$, and evaluates a message dependent function at a fixed value. This can lead to some weaknesses, indeed in uses of SNOW 3G in LTE the fixed value is alterred at each invocation in a highly similar construction. Without this fix, there is a growing body of work examining weaknesses of the construction, e.g. [143, 289, 303]. Due to these potential issues use of GMAC outside of GCM mode is we leave in the legacy only division. See the entry on GCM mode below for further commentary.

4.2.3 Hash Function Based MACs

HMAC

The HMAC scheme[1] was introduced in [29] and standardized in [172, 207]. The construction is based on an underlying hash function which, itself, needs to have an iterative design. Provable security results for HMAC aim to establish that HMAC is a PRF [28, 29]. Interestingly, this can be done only relying on the pseudorandomness of the underlying hash-function and does not require collision-resistance [28]. In particular, this means that instantiations of HMAC with hash-functions that are not collision-resistant may still be reasonably secure, provided that the collision attacks do not yield distinguishing attacks against the psuedorandomness of the underlying hash function. HMAC-MD4 should therefore not be used while HMAC-SHA1, HMAC-MD5 are still choices for which forgeries cannot be made. Conservative instantiations should consider HMAC-SHA2 and HMAC-SHA3.

UMAC

UMAC was introduced in [49] and specified in [209]. The scheme has provable security guarantees [49]. The scheme uses internally a universal-hash function for which the computation can be paralellized which in turn allows for efficient implementations with high throughput. The scheme requires a nonce for each application; one should ensure that the input nonces do not repeat. Rekeying should occur after 2^{64} applications. Due to analysis by Hanschuh and Preneel [143], the 32-bit output version resilts in a full key recovery after a few chosen texts and 2^{40} verifications. This implies one also needs to limit the number of verifications, irrespective of nonce reuse.

4.3 Authenticated Encryption

An authenticated encryption (AE) scheme aims to provide a stronger form of confidentiality than that achieved by the IND-CPA modes of operation considered earlier. In particular an AE scheme

[1]The standard ISO 9797-2 specifies three closely related schemes that can be seen as instantiations of NMAC with different parameters

provides both confidentiality (IND-CPA) and ciphertext integrity (INT-CTXT), both of which together imply security for Authenticated Encryption (a stronger notion than standard IND-CCA). An authenticated encryption scheme which is for one-time use only is often called a Data Encapsulation Mechanism (DEM) . .

4.3.1 Encrypt-then-MAC (and variants)

Encrypt-then-MAC is probably the simplest mechanism to construct an authenticated encryption scheme. The security of the method was studied in [32], where the benefits over other techniques are discussed. The main disadvantage when using Encrypt-then-MAC is that it is a two pass process.

Usage of Encrypt-then-MAC with CBC mode as the encryption scheme (with zero-IV) and CBC-MAC as the message authentication code is a common DEM for use with public key KEMs to produce public key encryption schemes. Use of zero-IV in non DEM (i.e. non one-time applications) is not recommended, due to the basic requirement of probabilistic encryption in most applications.

Other related techniques, such as Encrypt-and-MAC or MAC-then-Encrypt, in general *should not* be used as various real world attacks have been implemented on systems which use these insecure variants; for example SSL/TLS uses MAC-then-Encrypt and in such a configuration suffers from an attack [13]. Methods such as MAC-then-Encrypt can be shown to be secure in specific environments and with specific components (i.e. specific underlying IND-CPA encryption scheme and specific underlying MAC), however the probability of an error being made in the choice, implementation or application are too large to allow recommendation.

4.3.2 OCB

Offset Codebook (OCB) mode [168] was proposed by Rogaway et al. [299]. The mode's design is based on Jutla's authenticated encryption mode, IAPM. OCB mode is provably secure assuming the underlying block cipher is secure. OCB mode is a one-pass mode of operation making it highly efficient. Only one block cipher call is necessary for each plaintext block, (with an additional two calls needed to complete the whole encryption process).

The adoption of OCB mode has been hindered due to two U.S. patents. As of January 2013, the author has stated that OCB mode is free for software usage under an GNU General Public License, and for other non-open-source software under a non-military license [298].

4.3.3 CCM

CCM mode [259] was proposed in [342] and essentially combines CTR mode with CBC-MAC, using the same block cipher and key. The mode is defined only for 128-bit block ciphers and is used in 802.11i. A proof of security was given in [179], and a critique has been given in [300].

The main drawback of CCM mode comes from its inefficiency. It is a two-pass method, meaning that each plaintext block implies two block cipher calls. Secondly, the mode is not "online", as a result the whole plaintext must be known before encryption can be performed. An online scheme allows encryption to be perform on-the-fly as and when plaintext blocks are available. For this reason (amongst others) CCM mode has in some sense been superseded by EAX mode.

4.3.4 EAX

EAX mode [168] was presented in [36], where an associated proof of security was also given. It is very similar to CCM mode, also being a two-pass method based on CTR mode and CBC-MAC but with the advantage that both encryption and decryption can be performed in an online manner.

4.3.5 CWC

Carter-Wegman + Counter (CWC) mode was designed by Kohno, Viega and Whiting [205]. As the name suggests it combines a Carter-Wegman MAC, to achieve authenticity, with CTR mode encryption, to achieve privacy. It is provably secure assuming the IV is a nonce and the underlying block cipher is secure. When considering whether to standardise CWC mode or GCM, NIST ultimately chose GCM. As a result GCM is much more widely used and studied.

4.3.6 GCM

Galois/Counter Mode (GCM) [260] was designed by McGrew and Viega [236, 237] as an improvement to CWC mode. It again combines Counter mode with a Carter-Wegman MAC (i.e. GMAC), whose underlying hash function is based on polynomials over the finite field $GF(2^{128})$. GCM is widely used and is recommended as an option in the IETF RFCs for IPsec, SSH and TLS. The mode is online, is fully parallelisable and its design facilitates efficient implementations in hardware.

GCM is provably secure [174] assuming that the IV is a nonce and the underlying block cipher is secure. Note that repeating IVs lead to key recovery attacks [143]. Joux [181] demonstrated a problem in the NIST specification of GCM when non-default length IVs are used. Ferguson's [180] critique highlights a security weakness when short authentication tags are used. To prevent attacks based on short tags it is recommended that authentication tags have length at least 96 bits. Furthermore it is recommended that the length of nonces is fixed at 96 bits. Saarinen [303] raises the issues of weak keys which may lead to cycling attacks. For messages which are not "too large" such attacks are not a concern. The work of [289] presents an algebraic analysis which demonstrates even more weak keys.

4.4 Key Derivation Functions

Key Derivation Functions (KDFs) are used to derive cryptographic keys from secret shared random strings. For example they are used to derive keys for use in authenticated encryption schemes from a secret shared random string which is determined via a public key encapsulation. In security proofs they are often modelled as random oracles; but simply instantiating them with a vanilla hash function is not to be recommended (despite this being common practice in academic papers). Thus specific KDFs are designed for use in various situations. Often they take additional input of a shared info field, which is not necessarily secret. We summarize the constructions in Table 4.3, where the column "Building Block" refers to the underlying primitive used to create the KDF primitive.

4.4.1 NIST-800-108-KDF

NIST-SP800-108 [257] defines a family of KDFs based on psuedo-random-functions PRFs. These KDFs can produce arbitrary length output and they are formed by repeated application of the PRF. One variant (Counter mode) applies the PRF with the input secret string as key, to an input consisting of a counter and auxiliary data; one variant (Feedback mode) does the same but also takes as input in each round the output of the previous round. The final double pipelined mode uses two iterations of the same PRF (with the same key in each iteration), but the output of the first iteration (working in a feedback mode) is passed as input into the second iteration; with the second iteration forming the output. The standard does not define how any key material is turned into a key for the PRF, but this is addressed in NIST-SP800-56C [264].

4.4.2 X9.63-KDF

This KDF is defined in the ANSI standard X9.63 [20] and was specifically designed in that standard for use with elliptic curve derived keys; although this is not important for its application. The KDF works by repeatedly hashing the concatenation of the shared random string, a counter and the shared info. The KDF is secure in the random oracle model.

4.4.3 NIST-800-56-KDFs

A variant of the X9.63-KDF is defined in NIST-SP800-56A/B, [262, 263]. The main distinction being the hash function is repeatedly applied to the concatenation of the counter, the shared random string and the shared info (i.e. a different order is used).

In NIST-SP800-56C [264] a different KDF is defined which uses a MAC function application to obtain the derived key; with a publicly known parameter (or salt value) used as the key to the MAC. This KDF has stronger security guarantees than the hash function based KDFs (for example one does not need a proof in the random oracle model). However, the output length is limited to the output length of the MAC, which can be problematic when deriving secret keys for use in authenticated encryption schemes requiring double length keys (e.g. Encrypt-then-MAC). For this reason the standard also specifies a key expansion methodology based on NIST-800-108 [257], which takes the same MAC function used in the KDF, and then uses the output of the KDF as the key to the MAC function so as to define a PRF.

4.4.4 IKE-v1-KDF and IKE-v2-KDF

This is the KDF specified in [144] and [191] for use with Diffie–Hellman agreed keys (and others) derived in the IKE sub-protocol of IPsec. The methods use a HMAC as a PRF. In both variants HMAC is first used to extract randomness from the shared random value (i.e. a Diffie–Hellman secret), and then HMAC is used again to derive the actual key material. The IETF considers the Version 1 of the KDF to be obsolete.

4.4.5 TLS-KDF

This is the KDF defined for use in TLS, it is defined in [93] and [50]. In the TLS v1.0 and v1.1 versions of the KDF, HMAC-SHA1 and HMAC-MD5 are used as KDFs and their outputs are then exclusive-or'd together; producing a PRF sometimes called *HMAC-MD5/HMAC-SHA1*. In TLS v1.2 the PRF is simply HMAC instantiated with SHA-2. In both cases the underlying PRF is used to both extract randomness and for key expansion.

4.5 Generalities on Public Key Schemes

Before using a public key scheme there are some basic operations which need to be performed. We recap on these here as an aid memoir for the reader, but do not discuss them in much extra detail.

- **Certification:** Public keys almost always need to be certified in some way; i.e. a cryptographic binding needs to be established between the public key and the identity of the user claiming to own that key. Such certification usually comes in the form of a digital certificate, produced using a recommended signing algorithm. This is not needed for the identity based schemes considered later.

- **Domain Parameter Validation:** Some schemes, such as those based on discrete logarithms, share a set of a parameters across a number of users; these are often called Domain Parameters. Before using such a set of domain parameters a user needs to validate them to be secure, i.e. to meet the security level that the user is expecting. To ease this concern it is common to select domain parameters which have been specified in a well respected standards document.

- **Public Key Validation:** In many schemes and protocols long term or ephemeral public keys need to be validated. By this we mean that the data being received actually corresponds to a potentially valid public key (and not a potentially weak key). For example this could consist of checking whether a received elliptic curve point actually is a point on the given curve, or does not lie in a small subgroup. These checks are very important for security but often are skipped in descriptions of protocols and academic treatments.

4.6 Public Key Encryption

Public key encryption schemes are rarely used to actually encrypt messages, they are usually used to encrypt a symmetric key for future bulk encryption. Of the schemes considered below only RSA-PKCS# 1 v1.5 and RSA-OAEP can be considered as traditional public key encryption algorithms. Non-KEM based applications should only be used when encrypting small amounts of data, and in this case only RSA-OAEP is secure.

4.6.1 RSA-PKCS# 1 v1.5

This encryption method defined in [281,282] has no modern security proof, although it is used in the SSL/TLS protocol extensively. A chosen ciphertext reaction attack [52] can be applied, although the operation of the encryption scheme within SSL/TLS has been modified to mitigate against this specific attack. The weak form of padding can also be exploited in other attacks if related messages and/or a low public exponent are used [80,83,146]. This method of encryption is not recommended for any applications, bar the specific use (for legacy reasons) in SSL/TLS.

4.6.2 RSA-OAEP

Defined in [282], and first presented in [35], this is the preferred method of using the RSA primitive to encrypt a *small* message. It is known to be provably secure in the random oracle model [130]. A decryption failure oracle attack is possible [227] if implementations are not careful in uniform error reporting/constant timing. Security is proved in the random oracle model, i.e. under the assumption that the hash functions used in the scheme behave as random oracles. It is recommended that the hash functions used in the scheme be implemented with SHA-1 for legacy applications and SHA-2/SHA-3 for future applications.

4.7 Hybrid Encryption

The combination of a Key Encapsulation Mechanism (KEM) with a Data Encryption Mechanism (DEM) (both secure in the sense of IND-CCA) results in a secure (i.e. IND-CCA) public key encryption algorithm; and is referred to as a hybrid cipher. This is the preferred method for performing public key encryption of data, and is often called the KEM-DEM paradigm.

Various standards specify the precise DEM to be used with a specific KEM. So for example ECIES can refer to a standardized scheme in which a specific choice of DEM is mandated for use with ECIES-KEM. In this document we allow *any* DEM to be used with *any* KEM, the exact choice is left to the user. The precise analysis depends on the security level (legacy or future) we assign to the DEM and the constituent parts; as well as the precise instantiation of the underlying public key primitive.

4.7.1 RSA-KEM

Defined in [166], this Key Encapsulation Method takes a random element $m \in \mathbb{Z}/N\mathbb{Z}$ and encrypts it using the RSA function. The resulting ciphertext is the encapsulation of a key. The output key is given by applying a KDF to m, so as to obtain a key in $\{0,1\}^k$. The scheme is secure in the random oracle model (modeling the KDF as a random oracle), with very good security guarantees [147,319]. It is recommended that the KDF used in the scheme be one of the recommendations from Section 4.4.

4.7.2 PSEC-KEM

This scheme is defined in [166]. Again when modeling the KDF as a random oracle, this scheme is provable secure, assuming the computational Diffie–Hellman problem is hard in the group under which the scheme is instantiated. Whilst this gives a stronger security guarantee than ECIES-KEM below, in that security is not based on gap Diffie–Hellman, the latter scheme is often preferred due to performance considerations. Again it is recommended that the KDF used in the scheme be one of the recommendations from Section 4.4.

4.7.3 ECIES-KEM

This is the discrete logarithm based encryption scheme of choice. Defined in [20, 166, 314], the scheme is secure assuming the KDF is modelled as a random oracle. However, this guarantee is requires one to assume the gap Diffie–Hellman is hard (which holds in general elliptic curve groups but sometimes not in pairing groups). Earlier versions of standards defining ECIES had issues related to how the KDF was applied, producing a form of *benign malleability*, which although not a practical security weakness did provide unwelcome features of the scheme. Again it is recommended that the KDF used in the scheme be one of the recommendations from Section 4.4.

4.8 Public Key Signatures

4.8.1 RSA-PKCS# 1 v1.5

Defined in [281, 282] this scheme has no security proof, nor any advantages over other RSA based schemes such as RSA-PSS below, however it is widely deployed. As such we do not recommend use beyond legacy systems.

4.8.2 RSA-PSS

This scheme, defined in [282], can be shown to be UF-CMA secure in the random oracle model [178]. It is used in a number of places including e-passports.

4.8.3 RSA-FDH

The RSA-FDH scheme hashes the message to the group $\mathbb{Z}/N\mathbb{Z}$ and then applies the RSA (decryption) function to the output. The scheme has strong provable security guarantees [81,82,187], but is not recommended for use in practice due to the difficulty of defining a suitably strong hash function with codomain the group $\mathbb{Z}/N\mathbb{Z}$. Thus whilst conceptually simple and appealing the scheme is not practically deployable.

One way to instantiate the hash function for an $\ell(N)$ bit modulus would be to use a hash function with an output length of more than $2 \cdot \ell(N)$ bits, and then take the output of this hash function modulo N so as to obtain the pre-signature. This means the full domain of the RSA function will be utilized with very little statistical bias in the distribution obtained. This should be compared with ISO's DS3 below.

4.8.4 ISO 9796-2 RSA Based Mechanisms

ISO 9796-2 [170] defined three different RSA signature padding schemes called Digital Signature 1, Digital Signature 2 and Digital Signature 3. Each scheme supports either full or partial message recovery (depending of course on the length of the message). We shall refer to these as DS1, DS2 and DS3.

Variant DS1 essentially RSA encrypts a padded version of the message along with a hash of the message. This variant has been attacked by Coron et al [84, 85] which reduced breaking the padding scheme from 2^{80} operations to 2^{61} operations. Using a number of implementation tricks the authors of [85] manage to produce forgeries in a matter of days utilizing a small number of machines. Thus this variant should no longer be considered secure.

Variant DS2 is a standardized version of RSA-PSS, but in a variant which allows partial message recovery. All comments associated to RSA-PSS apply to variant DS2.

Variant DS3 is defined by taking DS2 and reducing the randomization parameter to length zero. This results in a deterministic signatures scheme which is "very close" to RSA-FDH, but for which the full RSA domain is not used to produce signatures. The fact that a hash image is not taken into the full group $\mathbb{Z}/N\mathbb{Z}$ means the security proof for RSA-FDH does not apply. We therefore do not recommend the use of DS3 for future applications.

4.8.5 (EC)DSA

The Digital Signature Algorithm (DSA) and its elliptic curve variant (ECDSA) is widely standardized [19, 314]; and there exists a number of variants including the German DSA (GDSA) [151, 164], the Korean DSA (KDSA) [164, 331] and the Russian DSA (RDSA) [136, 165]. The basic construct is to produce an ephemeral public key (the first part of the signature component), then hash the message to an element in $\mathbb{Z}/q\mathbb{Z}$, and finally to combine the hashed message, the static secret and the long term secret in a "signing equation" to produce the second part of the signature.

All (EC)DSA variants have weak provable security guarantees; whilst some proofs do exist they are in less well understood models (such as the generic group), for example [63]. The reason for this is that the hash function is only applied to the message and not the combination of the message and the ephemeral public key.

All (EC)DSA variants also suffer from lattice attacks against poor ephemeral secret generation [155, 255, 256]. A method to prevent this, proposed in [294] but known to be "folklore", is derive the ephemeral secret key by applying a PRF (with a default key) to a message containing the static secret key and the message to be signed.

4.8.6 PV Signatures

ISO 14888-3 [164] defined a variant of DSA signatures (exactly the same signing equation as for DSA), but with the hash function computed on the message and the ephemeral key. This scheme is due to Pointcheval and Vaudeney [284]. The signatures can be shown to be provably secure in the random oracle model, and so have much of the benefits of Schnorr signatures. However Schnorr signatures have a simpler to implement signing equation (no need for any modular inversions). Whilst only defined in the finite field setting in ISO 14888-3, the signatures can trivially be extended to the elliptic curve situation.

Just like (EC)DSA signatures, PV signatures suffer from issues related to poor randomness in the ephemeral secret key. Thus the defenses proposed for (EC)DSA signatures should also be applied to PV signatures.

4.8.7 (EC)Schnorr

Schnorr signatures [313], standardized in [165], are like (EC)DSA signatures with two key differences; firstly the signing equation is simpler (allowing for some optimizations) and secondly the hash function is applied to the concatenation of the message and the ephemeral key. This last property means that Schnorr signatures can be proved UF-CMA secure in the random oracle model [283]. There is also a proof in the generic group model [254]. We believe Schnorr signatures are to be preferred over DSA style signatures for future applications.

Just like (EC)DSA signatures, Schnorr signatures suffer from issues related to poor randomness in the ephemeral secret key. Thus the defenses proposed for (EC)DSA signatures should also be applied to Schnorr signatures.

4.9 Identity Based Encryption/KEMs

4.9.1 BF

The Boneh–Franklin IBE scheme [58, 59] is known to be fully (ID-IND-CCA) secure in the random oracle model and is presented in the IEEE 1363.3 standard [160]. The scheme is not as efficient as the following two schemes, and it does not scale well with increased security parameters; thus it is only recommended for legacy use. The underlying construction can also be used in a KEM mode.

4.9.2 BB

The Boneh–Boyen IBE scheme [56] is secure in the standard model under the decision Bilinear Diffie–Hellman assumption, but only in a weak model of selective ID security. However, the scheme, as presented in the IEEE 1363.3 standard [160], hashes the identities before executing the main BB scheme. The resulting scheme is therefore fully secure in the random oracle model. The scheme is efficient, including at high security levels, and has a number of (technical) advantages when compared to other schemes.

4.9.3 SK

The Sakai–Kasahara key construction is known to be fully secure in the random oracle model, and at the same curve/field size outperforms the prior two schemes. The constructions comes as an encryption scheme [73] and a KEM construction [74], and is also defined in the IEEE 1363.3 standard [160]. The main concern on using this scheme is due to the underlying hard problem (the q-bilinear Diffie–Hellman inversion problem) not being as hard as the underlying hard problem of the other schemes. This concern arises from a series of results, initiating with those of Cheon [75], on q-style assumptions.

Chapter 5

Protocols

Our choice of protocols to cover is mainly dictated by what we feel is of most interest to the reader. However, there has been much less analysis of protocols, such as those in this chapter, compared to the primitives and schemes presented in previous chapters. Thus much of what we discuss in this chapter can be seen as more likely to change as the research community shifts its focus to analyzing protocols over the coming decade. The reader will hopefully also see by our analysis that most of the deployed protocols which can be used by a naive user, are in fact either incredibly complex to install in a manner which we would deam secure, or are in fact insecure with respect to modern cryptographic standards.

We divide our discussion on protocols into general protocols for which there is some choice as to using them and application specific protocols for which there is no choice as to usage. For example application developers often choose to use TLS as a means to securing a channel between applications, and the various parameters can then be selected by the developer. On the other hand one is forced to use UMTS/LTE if one wishes to use a standard mobile phone, with provider mandated parameters and algorithms. In addition the use of UMTS/LTE outside this application is very limited. Clearly this division is not complete, as in some sense one is also forced to use TLS when accessing secure web sites, but the usage of TLS is much wider than that.

5.1 General Protocols

5.1.1 Key Agreement

We first discuss key agreement, since this is the one areas in which there has been a rigourous analysis of protocols; with concrete security definitions being given. Despite this the situation in relation to how these security definitions map onto real world protocols and their usages is still in a state of flux.

The NIST standard [262] (resp. [263]) and the ANSI standards [17,20] (resp. [18]) define methods for general key agreement using discrete logarithm based systems (resp. factoring based systems).

The standard [262] introduces a nice taxonomy for such schemes with the notation $C(a, b)$, where $a, b \in \{0, 1, 2\}$. The number a refers to how many of the two parties contribute ephemeral keys to the calculation and the number b refers to how many of the two parties are authenticated by long term public/private key pairs. For example, traditional non-authenticated Diffie–Hellman is denoted as a $C(2, 0)$ scheme, where as traditional MQV [212] is denoted as a $C(2, 2)$ scheme. The standards also provide various mechanisms for key confirmation.

The security of key agreement schemes is somewhat complicated. The traditional security models of Bellare, Rogaway, et al base security on indistinguishablity of keys [33, 34, 51, 68]. This property is often not satisfied by real world protocols, and in particular by protocols using key confirmation. This issue has started to be treated in a number of works focusing on the TLS protocol (see below). Also discussion of the notion of one-sided authentication in key agreement has only recently started in the academic literature [65, 208]. Thus many of the options in these standards cannot be said to have fully elaborated proofs of security which are applicable in general situations.

The precise choice of which key agreement scheme to use is therefore highly dictated by the underlying application. However, the current document can be used to determine key sizes (for the underlying factoring and discrete logarithm based primitives) as well as the key derivation functions, MAC functions and any other basic cryptographic components used.

5.1.2 TLS

The TLS protocol is primarily aimed at securing traffic between an unauthenticated web browser and an authenticated web site, although the protocol is now often used in other applications due in part to the availability (and ease of use) of a variety of libraries implementing TLS. The TLS protocol suite aims to provide a confidential channel rather than simply a key agreement protocol as discussed before. The key agreement phase has now been fairly thoroughly analyzed in a variety of works [64, 175, 176, 208, 247]. A major issue in these analyses is the use of the derived key during key confirmation via the `FINISHED` messages. Care must be taken in long term key generation as a number of TLS implementations have been shown to be weak due to poor random number generation [148].

The key agreement phase runs in one of two main modes: either RSA-based key transport or Diffie–Hellman key exchange (an option also exists for pre-shared keys). The RSA key transport methodology uses RSA-PKCS#1 v1.5, which as discussed previously in Section 4.6.1 is not considered secure in a modern sense. However, the use of this key transport methodology has been specifically patched in TLS to avoid the attack described in [52], and a formal security analysis supporting this approach in the TLS context can be found in [208]. In both modes the output of the key agreement phase is a so-called pre-master secret.

During the key agreement phase the key to use in the transport layer is derived from the agreed pre-master secret. This derivation occurs in one of two ways, depending on whether TLS 1.2 [95] is used or whether an earlier standard is used (TLS 1.0 [94] and TLS 1.1 [94]). As discussed in Section 4.4.5, the use of TLS-v1.1-KDF should only be used for legacy applications, with the TLS-v1.2-KDF variant being considered suitable for future applications.

The record layer, i.e. the layer in which actual encrypted messages are sent and received, has received extensive analysis. In TLS 1.0 and TLS 1.1 the two choices are either MAC-then-Encode-then-Encrypt using a block cipher in CBC mode or the use of MAC-then-Encrypt using the RC4 stream cipher. Both these forms of the record layer have been shown to be problematic [12, 13, 71, 273, 335]. The main problems here are that the MAC-then-Encode-then-Encrypt construction used in TLS is difficult to implement securely (and hard to provide positive security results about), and that RC4 is, by modern standards, a weak stream cipher. These issues are partially corrected in TLS 1.2 [95] by adding support for Authenticated Encryption, and with GCM mode and CCM mode for TLS being specified in [305] and [234], respectively. Other recent attacks include those by Duong and Rizzo, known as BEAST [99] and CRIME [100]. BEAST exploits the use of chained IVs in CBC mode in TLS 1.0, and CRIME takes advantage of information leakage from the optional use of data compression in TLS.

Given the above discussion it is hard to recommend that TLS 1.0 and TLS 1.1 be used in any new application, and phasing out their use in legacy applications is recommended. It would appear that TLS 1.2 is sufficient for future applications. However, there are currently very few implementations of clients, servers, or libraries which support TLS 1.2.

A complete list of ciphersuites for TLS is listed at the website `http://www.iana.org/assignments/tls-parameters/tls-parameters.xml`. If following the recommendations of this document, the restrictions on the ciphersuites to conform to our future recommendations means this relatively large list becomes relatively small. Looking at the record layer protocol (i.e. the algorithms to encrypt the actual data), we see that only the use of Camellia and AES are recommended within a mode such as GCM or CCM.

For the handshake, key agreement, part of the protocol the principle issue is that the RSA signing algorithm in TLS 1.2 is RSA-PKCS# 1 v1.5. As explained in Section 4.8.1 we do not recommend the use of this signature scheme in future systems. The RSA key transport mechanism is itself based on RSA-PKCS# 1 v1.5, which we again pointed out in Section 4.6.1 has no formal proof of security, and which cannot be recommended beyond its use in legacy systems. However, a recent analysis [208] does show that RSA-PKCS# 1 v1.5 for key transport in TLS can be made secure under a sufficiently strong number theoretic assumption and in the Random Oracle Model. Considering the discrete logarithm or elliptic curve variants, one finds that the situation is little better. The required signature algorithm here is (EC)DSA, which also has no proof of security, bar in the generic group model for the elliptic curve variant. See Section 4.8.5 for more details. Thus for the key negotiation phase one is left to rely on cryptographic schemes which we only recommend for legacy use.

This means at the time of writing we would only recommend the following cipher suites, for future use within TLS

- `⋆_WITH_Camellia\index{Camellia}_128_GCM_SHA256`,

- `⋆_WITH_AES_128_GCM_SHA256`,
- `⋆_WITH_Camellia\index{Camellia}_256_GCM_SHA384`,
- `⋆_WITH_AES_256_GCM_SHA384`,
- `⋆_WITH_AES_128_CCM`,
- `⋆_WITH_AES_128_CCM_8`,
- `⋆_WITH_AES_256_CCM`,
- `⋆_WITH_AES_256_CCM_8`.

where ⋆ is suffix denoting the underlying key exchange primitive.

5.1.3 IPsec

IPsec is designed to provide security at the IP network layer of the TCP/IP protocol stack. This differs from protocols such as TLS and SSH which provide security at higher layers such as the application layer. This is advantageous since no re-engineering of the applications is required to benefit from the security IPsec provides. The main use of IPsec has been to create virtual private networks (VPNs) which facilitates secure communication over an untrusted network such as the Internet. The protocol was originally standardised in a collection of RFCs in 1995 and their third incarnation can be found in RFCs 4301–4309 [103, 152, 154, 191, 194–197, 311]. For a more complete description of the cryptography in the IPsec standards we refer the reader to the survey by Paterson [271].

The IPsec protocols can be deployed in two basic modes: tunnel and transport. In tunnel mode cryptographic protection is provided for entire IP packets. In essence, a whole packet (plus security fields) is treated as the new payload of an outer IP packet, with its own header, called the outer header. The original, or inner, IP packet is said to be encapsulated within the outer IP packet. In tunnel mode, IPsec processing is typically performed at security gateways (e.g. perimeter firewalls or routers) on behalf of endpoint hosts. By contrast, in transport mode, the header of the original packet itself is preserved, some security fields are inserted, and the payload together with some header fields undergo cryptographic processing. Transport mode is typically used when end-to-end security services are needed, and provides protection mostly for the packet payload. In either mode, one can think of the IPsec implementation as intercepting normal IP packets and performing processing on them before passing them on (to the network interface layer in the case of outbound processing, or to the upper layers in the case of inbound processing).

Each IPsec implementation contains a Security Policy Database (SPD), each entry of which defines processing rules for certain types of traffic. Each entry in the SPD points to one or more Security Associations (SAs) (or the need to establish new SAs). The SAs contain (amongst other information) cryptographic keys, initialisation vectors and anti-replay counters, all of which must be initialised and shared between appropriate parties securely. This can be solved manually, and such an approach works well for small-scale deployments for testing purposes. However, for larger scale and more robust use of IPsec, an automated method is needed. The Internet Key Exchange (IKE) Protocol provides the preferred method for SA negotiation and associated cryptographic parameter establishment. The latest version of IKE, named IKEv2 [192], provides a flexible set of methods for authentication and establishment of keys and other parameters, supporting both asymmetric and symmetric cryptographic methods. There were initially two Diffie–Hellman Groups defined for use in IKEv2 [192, Appendix B], one with a 768-bit modulus the other with 1024-bit modulus. Further DH groups are defined in RFC3526 [200] of sizes 1536, 2048, 3072, 4096, 6144 and 8192 bits. Elliptic Curve groups are defined in RFC 5903 [129] with sizes of 256, 384 and 521 bits. RFC5114 [218] defines an additional 8 groups. Based on Section 3.5 we recommend for future use a group size of at least 3072 bits, and 256 bits in the case of elliptic curve groups. For key derivation, as discussed in Section 4.4.4, the use of IKE-v1-KDF should only be used for legacy applications, with the IKE-v2-KDF variant being considered suitable for future applications.

There are two main IPsec protocols which specify the actual cryptographic processing applied to packets. These are called Authentication Header (AH) and Encapsulating Security Payload (ESP).

AH provides integrity protection, data origin authentication and anti-replay services for packets through the application of MAC algorithms and the inclusion of sequence numbers in packets. There are a number of MAC algorithms defined for use with IPsec. These include HMAC (with MD5 [224], SHA-1 [225] or SHA-2 [193]), GMAC [235] and XCBC (a CBC-MAC variant) [127]. Based on earlier chapters we only recommend HMAC with SHA-2 for future use.

ESP provides similar services to AH (though the coverage of its optional integrity protection feature is more limited) and in addition provides confidentiality and traffic flow confidentiality services through symmetric key encryption and variable length padding of packets. ESP allows both encryption-only and authenticated encryption modes. The attacks we shall mention in the following paragraph demonstrate the encryption-only modes should *not* be used. ESP must therefore always be configured with some form of integrity protection. The encryption algorithms on offer are CBC mode (with either 3DES [278], AES [126] or Camellia [190]), CTR mode (with either AES [153] or Camellia [190]). Of these algorithms we would only recommend CTR mode and stress it must be combined with a MAC algorithm. Further options for authentication encryption are provided by the combined algorithms CCM (with either AES [154] or Camellia [190])and GCM with AES [154].

An initial analysis of the IPsec standards was performed by Ferguson and Schneier [121]. This was followed by Bellovin [38] who found a number of attacks against encryption-only ESP. Practical attacks were demonstrated by Paterson and Yau [277] against the Linux implementation of IPsec where encryption-only ESP was operating in tunnel mode. By adapting the padding oracle attack of Vaudenay [335], Degabriele and Paterson were then able to break standards-compliant implementations of IPsec [91] with practical complexities. These attacks were against encryption-only ESP using CBC mode and operating in either tunnel or transport mode. From these attacks, the need to use some form of integrity protection in IPsec is evident. It is therefore recommended that encryption-only ESP *not* be used. A further set of attacks by Degabriele and Paterson [92] breaks IPsec when it is implemented in any MAC-then-Encrypt configuration (for example, if AH in transport mode is used prior to encryption-only ESP in tunnel mode). On the other hand, no attacks are known if ESP is followed by AH, or if ESP's innate integrity protection feature is used. To conclude, we reiterate that ESP should always be used with some form of integrity protection, and that care is needed to ensure an appropriate form of integrity protection is provided.

A close to complete list of ciphersuites for IPsec is listed at the website `http://www.iana.org/assignments/isakmp-registry/isakmp-registry.xml`. Based on the information in this document we would only recommend the following algorithms for future use within IPsec:

If only authentication is required then either AH or ESP may be used with one of the following MAC algorithms as defined in RFC4868 [193].

- `HMAC-SHA2-256`,
- `HMAC-SHA2-384`,
- `HMAC-SHA2-512`,

If confidentiality is required then ESP should be used by combining one of the following encryption algorithms with one of the MAC algorithms described above.

- `AES-CTR`,
- `CAMELLIA-CTR`,

Alternatively one of the following combined authenticated encryption modes may be used:

- `AES-CCM_`$\star$,
- `CAMELLIA-CCM_`$\star$,
- `AES-GCM_`$\star$,

Here $\star$ denotes the size (in bytes) of the integrity check value (ICV) and we recommend choosing either 12 or 16.

5.1.4 SSH

Secure Shell (SSH) was originally design as a replacement for insecure remote shell protocols such as telnet. It has now become a more general purpose tool that is used to provide a secure channel between two networked computers for applications such as secure file transfer. SSHv2 was standardised in a collection of RFCs [347–349] in 2006. The originally version, SSHv1 has several design flaws and should no longer be used. OpenSSH [1] is the most widely used implementation of the protocol. As of 2008 it accounted for more than 80% of all implementations. The transport layer of SSH [349] is responsible for the initial key-exchange, server authentication and, confidentiality and integrity of messages sent on the channel.

The key-exchange protocol is based on Diffie–Hellman and host authentication is provided by combining this with a signature. Client authentication is also possible but defined in a separate RFC [347]. Methods for authenticating the client are either using a password, public-key cryptography (DSA, RSA, X.509), an "interactive-keyboard" challenge-response method [87] or the GSSAPI [221] which allows the use of external mechanisms such as Kerberos. Support for the key-exchange methods, `diffie-hellman-group1-sha1` and `diffie-hellman-group14-sha1` is mandated by the RFC [349]. These methods use the Oakley Group 1 (1024-bit prime field) and Oakley Group 14 (2048-bit prime field) [200]. RFC4419 [128] describes a key-exchange method for SSH that allows the server to propose new groups on which to perform the Diffie–Hellman key exchange with the client. RFC4432 [145] specifies a key-transport method for SSH based on 1024-bit and 2048-bit RSA. RFC5656 [323] defines introduces support for Elliptic-Curve Cryptography; detailing support for ECDH and ECMQV.

Williams [345] has performed an analysis of the key-exchange methods in SSH. It has been shown the six application keys (two IV keys, two encryption keys and two integrity keys) generated by the protocol and passed to the next stage of the SSH protocol are indistinguishable from random. The analysis assumes the server's public key is validated through a certificate from some secure public-key infrastructure. The author of [345] notes that if no such certificate is used, then the protocol is vulnerable to attack, unless the client has some other method of verifying the authenticity of a server's public key.

Once keys are established all message are then sent encrypted over the channel using the Binary-Packet Protocol (BPP) [349, Section 6]. This specifies an encryption scheme based on an Encode-then-Encrypt-and-MAC construction using a block cipher in CBC mode or the stream cipher RC4. The encode function specifies two length fields which must be prepended to messages prior to encryption and a padding scheme (for the case of CBC mode). The first length field specifies the total length of the packet and the second gives the total length of padding used. The specification recommends using CBC mode with chained IVs (the last block of the previous ciphertext becomes the IV of the following ciphertext). This has been shown to be insecure by Dai [89] and Rogaway [295]. Albrecht et al. [10] were able to perform plaintext-recovery attacks against SSH (when using CBC mode) by exploiting the use of encrypted length fields. As a result of these attacks we state that CBC mode *should not* be used. We note that OpenSSH Version 6.2 [1] supports a non-standard version of the BPP for use with CBC mode in an Encrypt-then-MAC construction where length fields are *not* encrypted but still authenticated. This style of construction would be secure against the Albrecht et al. attack.

A first formal security analysis of the SSH-BPP was performed by Bellare et al. [31]. As a result of the Albrecht et al. attacks this security analysis was proved to be incomplete and a further security analysis, which more closely matched actual implementations of SSH, was performed by Paterson and Watson [275]. They proved that the Encode-then-Encrypt-and-MAC construction utilizing counter mode encryption is secure against a large class of attacks including those of Albrecht et al.

We recommend counter mode as the best choice of available cipher in the Encode-then-Encrypt-and-MAC construction when combined with a secure MAC algorithm. The original choice of MAC algorithms specified in RFC4253 was limited to HMAC with either SHA-1 or MD5. We recommend neither of these hash functions for current use. RFC6668 [39] details the use of SHA-2 for HMAC.

In addition to the Encode-then-Encrypt-and-MAC construction confidentiality and integrity in SSH may also be provided by GCM encryption as specified in RFC5647 [161].

A complete list of ciphersuites for SSH is listed at the website `http://www.iana.org/assignments/ssh-parameters/ssh-parameters.xml`. Based on the recommendations of this document we would only recommend the following encryption and MAC algorithms, for future use within SSH:

- `aes128-ctr` with `hmac-sha2-256` or `hmac-sha2-512`
- `aes192-ctr` with `hmac-sha2-256` or `hmac-sha2-512`
- `aes256-ctr` with `hmac-sha2-256` or `hmac-sha2-512`
- `AEAD_AES_128_GCM`
- `AEAD_AES_256_GCM`

5.1.5 Kerberos

Kerberos is an authentication service which allows a client to authenticate his or herself to multiple services e.g. a file server or a printer. The system uses a trusted authentication server which will grant tickets to participating parties allowing them to prove their identity to each other. It is primarily based on symmetric-key primitives; the specific construction being derived from the Needham-Schroeder Protocol [252]. Public-key primitives, namely RSA signatures, may also be used during the initial authentication phase [350].

Kerberos was designed as part of project Athena at MIT during the 1980s [243]; the first three versions were not released publicly; Version 4 can therefore be viewed as the "original" release. The current version, Version 5 [253], fixed a number of security deficiencies present in its predecessor [37]. Version 4 required the use of DES; Version 5 expanded the possible ciphers and AES is now supported [291]. Additionally, Version 4 made use of a non-standard version of CBC mode called PCBC which has been shown to be insecure [204]. The encryption scheme used by Version 5 has been formally analyzed by Boldyreva and Kumar [54]. They first show that the Encode-then-Checksum-then-Encrypt construction defined in RFC3961 [292] does not meet the INT-CTXT notion of security. If a secure MAC algorithm is used for the checksum then this construction will be secure. Additionally, Boldyreva and Kumar analyse the Encode-then-Encrypt-and-MAC construction given in RFC3962 [291] and show this to be secure assuming the underlying primitives meet standard notions of security. The encryption scheme specified for use in Version 5 is CBC mode with ciphertext stealing using either DES, 3DES [292], AES [291] or Camellia [156] as the underlying blockcipher.

A complete list of ciphersuites for Kerberos is listed at the website `http://www.iana.org/assignments/kerberos-parameters/kerberos-parameters.xml`. At the time of writing we recommend the following ciphersuites for future use within Kerberos:

- `aes128-cts-hmac-sha1-96`
- `aes256-cts-hmac-sha1-96`
- `camellia128-cts-cmac`
- `camellia256-cts-cmac`

5.2 Application Specific Protocols

5.2.1 WEP/WPA

The protocols surveyed in this section are used to protect communication in wireless networks. We discuss their use in the setting where the device and the access point to which it connects have a shared key.

WEP (Wired Equivalent Privacy) is specified in the IEEE 802.11 standard [157]. The protocol is intended to offer private and authenticated communication. The protocol is symmetric key based (it uses either 40 bit, 104 bit, or 232 bit keys) and employs RC4 for privacy and CRC32 for authentication. Practical key-recovery attacks against the protocols have been devised [48,123,329] and the protocol is considered completely broken. The use of this protocol should be avoided. WEP has been deprecated by the IEEE.

WPA (Wi-Fi Protected Access) is a successor of WEP. It employs the Temporal Key Intergrity Protocol (TKIP) a stronger set of encryption and authentication algorithms; but TKIP has been deprecated by the IEEE. The protocol was intended as a temporary replacement for WPA capable of running on legacy hardware. The protocol fixes some of the desigen problems in WEP, but some attacks against TKIP have been found [142, 244, 246, 318, 328, 330]. A recent attack, [11], based on prior analysis of RC4 [12] in TLS, breaks this protocol. Thus uses should move to WPA2 as a matter of urgency.

WPA2 uses yet stronger primitives (see [158] for the latest version of the standard). It employs the Counter Cipher mode with Message Authentication Code Protocol (CCMP), an encryption scheme that uses AES in CCM mode (see Section 4.3.3) and offers both message privacy and message authentication. While some weaknesses in settings where WPA2 is used exist, no serious attacks are known against the protocol itself.

5.2.2 UMTS/LTE

The Universal Mobile Telecommunication System (UMTS) and its latest version called Long-Term Evolution (LTE) are standards for wireless communication in mobile phones and data terminals.

The standard is developed by the 3rd Generation Parternship Project (3GPP) and is now at version 10. The protocol is intended as a replacement for GSM. All technical specification documents referenced in this section are available at `www.3gpp.org`.

Very roughly, the protocol works in two phases, a key-establishment and authentication phase, and a data transmission phase. UMTS/LTE replaces the one-way authentication protocol used in GSM (which authenticates the mobile but not the network) with a stronger protocol called Authentication and Key Agreement (AKA). This is a three party protocol that involves a mobile station (MS) a serving network (SN) and the home environment (HE). Upon a succcesful execution of the protocol MS and SN have confirmed that they communicate with valid partners and establish a shared key. An additional design goal for the protocol is to protect the identity of the mobile station: an eavesdropper should not be able to determine weather the same mobile station was involved in two different runs of the protocol.

The key shared between MS and SN is used to implement a bi-directional secure channel between the two parties. Integrity and confidentiality are implemented (respectively) via algorithms UIA1 and UEA1 (in UMTS) [2] and UIA2 and UEA2 (in LTE) [3]. The algorithms have the same structure; the difference is determined by the underlying primitive: the Kasumi blockcipher [5] in UMTS and SNOW 3G streamcipher [4] in LTE.

There are no provable security guarantees for the protocol. The few published analysis for the protocol are mainly concerned with the anonymity guarantees [24, 206] and indicate that the protocol is susceptible to a number of attacks against MS privacy. Security of the channel established via UMTS/LTE had not been thoroughly analyzed. There are a few known theoretical weaknesses in Kasumi [41] and SNOW 3G [47, 199] but these do not seem to translate into attacks against the secure channel that they implement.

5.2.3 Bluetooth

Bluetooth is technology for exchanging data, securely, over short-distances between up to seven devices. The protocol stack for Bluetooth was originally standardized as IEEE802.15.1 standard, which is no longer maintained. The current development is overseen by the Bluetooth Special Interest Group.

This section discusses the cryptographic features of Bluetooth 2.1; the later versions (the latest is Bluetooth 4.0) are mainly concerned with improved bandwidth and power efficiency with little changes to the underlying cryptography.

Operating takes place in two stages. In the “pairing” stage, two Bluetooth devices agree on a pair of keys, an initialization key used for mutual authentication via a challenge response protocol based on HMAC-SHA-256; after authentication succeeds, the devices also agree on a link key for encrypting the traffic. Since Bluetooth 2.1 this stage is implemented with Elliptic Curve Diffie-Hellman (ECDH); depending on the capabilities of the devices involved, several mechanisms for providing protection against man-in-the-middle can be used. Data is encrypted in Bluetooth using streamcipher E0. Each packet is XORed with a keystream obtained by running the E0 algorithm on several inputs, one of which is the key link and another is a unique identifier.

The main weakness of Bluetooth is the pairing phase. Although stronger than in Bluetooth 1.0-2.0, pairing is still open to MITM attacks for devices without user input/output capabilities or other out-of-band communication means, or in configurations where a predefined PIN. As far as privacy of the communication goes, the few known theoretical attacks against E0 [122, 149] do not seem to impact privacy of messages. Message integrity protection is implemented with a cyclic redundancy code and is therefore minimal.

5.2.4 ZigBee

ZigBee is a radio communication standard which can be considered to operate mainly at lower power and ranges than Bluetooth. The key idea is to provide extended ranges by utilizing mesh networks of ZigBee connected devices. Bulk data encryption and authentication is based on the symmetric key mechanisms of IEEE 802.15.4 [159], and key management is implemented either by active key management with ZigBee-specific uses of ECDSA/ECDH or by predistribution of symmetric keys. The main algorithms are AES in CTR mode, an AES based CBC-MAC algorithm outputting either a 32-bit, 64-bit or 128-bit MAC value, or for combined authenticated encryption the use of AES in CCM mode, or a variant of CCM mode called CCM*. TLS support is provided with two mandatory cipher suites `TLS_PSK_WITH_AES_128_CCM_8` and `TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8`, these derive keying material either via symmetric pre-shared keys or via a elliptic curve Diffie–Hellman key exchange authenticated with ECDSA respectively. An optional suite of `TLS_DHE_RSA_WITH_AES_128_CCM_8` prepares the shared keying material via a finite field Diffie–Hellman exchange authenticated with RSA signatures.

5.2.5 EMV

The chip-and-pin bank/credit card system follows a specification defined by EMVCo. This document is mainly focused on cryptographic aspects and so we will restrict our discussion to the cryptographic components only; which are defined in “EMV Book 2” [112]. We therefore only mention in passing a number of systems security level issues observed by others [7, 8, 55, 97, 248].

Much of the existing EMV specification dates from before the advent of provable security; thus many of the mechanisms would not be considered cryptographically suitable for a new system. For example, the RSA based digital signature is DS1 from the standard ISO 9796-2 [170]; in a message recovery mode. As already explained in Section 4.8.4, this scheme suffers from a number of weaknesses, although none have been exploited to any significant effect in the EMV system. As a second example, the RSA encryption method (used to encrypt PIN blocks in some countries) is bespoke and offers no security guarantees. The only known analysis of this algorithm is in [322], which presents a Bleichenbacher-style attack against this specific usage. Another issue is that the card is allowed to use the same RSA private key for signing and encryption. This is exploited in [90] via another Bleichenbacher-style attack which converts the decryption oracle provided by the Bleichenbacher-style attack into a signing oracle; in turn, this can be used to forge EMV transaction certificates. It should be stated that neither of the above attacks have been shown to be exploitable "in the wild". Rather, they highlight potential problems with the current algorithm choices.

The symmetric key encryption schemes used in EMV are also slightly old fashioned. Two block ciphers are supported Triple DES and AES, with the underlying encryption method being CBC mode. The standard supports two MAC functions, AMAC for use with single DES and CMAC for use with AES.

EMVCo is currently engaged in the process of renewing their cryptographic specifications to bring them up to date. There has been a lot of work on defining elliptic curve based schemes for use in EMV. Some work has been done on analyzing the specific protocols and schemes being considered for use in the new specifications, for example see [65, 90].

Bibliography

[1] OpenSSH project. `http://www.openssh.org/`.

[2] Technical Specificaiton Group Services 3rd Generation Parternship Project and 3G Security System Aspects. Confidentiality and integrity algorithms UEA1 & UIA1. Document 1: UEA1 and UIA1 specifications.

[3] Technical Specificaiton Group Services 3rd Generation Parternship Project and 3G Security System Aspects. Confidentiality and integrity algorithms UEA2 & UIA2. Document 1: UEA2 and UIA2 specifications.

[4] Technical Specificaiton Group Services 3rd Generation Parternship Project and 3G Security System Aspects. Confidentiality and integrity algorithms UEA2 & UIA2. Document 2: SNOW 3G specification.

[5] Technical Specificaiton Group Services 3rd Generation Parternship Project and 3G Security System Aspects. Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification, v.3.1.1.

[6] Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors. *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*. Springer, 2007.

[7] Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Ross J. Anderson, and Ronald L. Rivest. On the security of the EMV secure messaging API (extended abstract). In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 5964 of *Lecture Notes in Computer Science*, pages 147–149. Springer, 2007.

[8] Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven J. Murdoch, Ross J. Anderson, and Ronald L. Rivest. Phish and chips. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 5087 of *Lecture Notes in Computer Science*, pages 40–48. Springer, 2006.

[9] Leonard M. Adleman. The function field sieve. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 1994.

[10] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext recovery attacks against SSH. In *IEEE Symposium on Security and Privacy*, pages 16–26. IEEE Computer Society, 2009.

[11] Nadhem AlFardan, Dan Bernstein, Kenny Paterson, Bertram Poettering, and Jacob Schuldt. Biases in the RC4 keystream (128 bit uniform vs. WPA/TKIP). `http://www.isg.rhul.ac.uk/tls/tkip_biases.pdf`, 2013.

[12] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C.N. Schuldt. On the security of RC4 in TLS. In *USENIX Security Symposium*. USENIX Association, 2013. To appear.

[13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2013.

[14] Ammar Alkassar, Alexander Geraldy, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. Optimized self-synchronizing mode of operation. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 78–91. Springer, 2001.

[15] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Knudsen [203], pages 83–107.

[16] ANSI X9.19. Financial institution retail message authentication. American National Standard Institute, 1996.

[17] ANSI X9.42. Agreement of symmetric keys using discrete logarithm cryptography. American National Standard Institute, 2005.

[18] ANSI X9.42. Key agreement and key transport using factoring-based cryptography. American National Standard Institute, 2005.

[19] ANSI X9.62. Public key cryptography for the financial services industry – The elliptic curve digital signature algorithm (ECDSA). American National Standard Institute, 2005.

[20] ANSI X9.63. Public key cryptography for the financial services industry – Key agreement and key transport using elliptic curve cryptography. American National Standard Institute, 2011.

[21] ANSSI. Référentiel Général de Sécurité, Annexe B1 Mécanismes cryptographiques : Régles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, Version 1.20 du 26 janvier 2010. `http://www.ssi.gouv.fr/IMG/pdf/RGS_B_1.pdf`, 2010.

[22] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In Matsui [230], pages 578–597.

[23] Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Halevi [140], pages 70–89.

[24] Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, and Mark Ryan. Formal analysis of UMTS privacy. *CoRR*, abs/1109.2066, 2011.

[25] Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2009.

[26] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, 2013.

[27] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. *J. Cryptology*, 21(3):392–429, 2008.

[28] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision resistance. In Dwork [101], pages 602–619.

[29] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[30] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.

[31] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241, 2004.

[32] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[33] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Preneel [286], pages 139–155.

[34] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[35] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[36] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Roy and Meier [302], pages 389–407.

[37] S. M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. *SIGCOMM Comput. Commun. Rev.*, 20(5):119–132, October 1990.

[38] Steven M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix Unix Security Symposium*, pages 1–16, 1996.

[39] D. Bider and M. Baushke. SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol. RFC 6668 (Proposed Standard), July 2012.

[40] Eli Biham and Yaniv Carmeli. Efficient reconstruction of RC4 keys from internal states. In Nyberg [267], pages 270–288.

[41] Eli Biham, Orr Dunkelman, and Nathan Keller. A related-key rectangle attack on the full KASUMI. In Roy [301], pages 443–461.

[42] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

[43] Alex Biryukov, editor. *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*. Springer, 2007.

[44] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In Gilbert [133], pages 299–319.

[45] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Matsui [230], pages 1–18.

[46] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory tradeoffs with multiple data. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer, 2005.

[47] Alex Biryukov, Deike Priemuth-Schmid, and Bin Zhang. Multiset collision attacks on reduced-round SNOW 3G and SNOW 3G $^{(+)}$. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 139–153, 2010.

[48] Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in WEP's coffin. In *IEEE Symposium on Security and Privacy*, pages 386–400. IEEE Computer Society, 2006.

[49] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In Wiener [344], pages 216–233.

[50] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492 (Informational), May 2006. Updated by RFC 5246.

[51] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.

[52] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[53] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Lee and Wang [213], pages 344–371.

[54] Alexandra Boldyreva and Virendra Kumar. Provable-security analysis of authenticated encryption in Kerberos. *IET Information Security*, 5(4):207–219, 2011.

[55] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei P. Skorobogatov, and Ross J. Anderson. Chip and skim: Cloning EMV cards with the pre-play attack. *CoRR*, abs/1209.2531, 2012.

[56] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[57] Dan Boneh and Glenn Durfee. Cryptanalysis of RSA with private key d less than $\mathrm{n}^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339–1349, 2000.

[58] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Kilian [198], pages 213–229.

[59] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[60] Joppe W. Bos and Marcelo E. Kaihara. Playstation 3 computing breaks 2^{60} barrier: 112-bit prime ECDLP solved. EPFL Laboratory for cryptologic algorithms - LACAL, 2009.

[61] Cyril Bouvier. Discrete logarithm in GF(2^{809}) with FFS. Post to NMBRTHRY@LISTSERV.NODAK.EDU, 2013.

[62] Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.

[63] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Des. Codes Cryptography*, 35(1):119–152, 2005.

[64] Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: Relaxed yet composable security notions for key exchange. *IACR Cryptology ePrint Archive*, 2012:242, 2012.

[65] Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. An analysis of the EMV channel establishment protocol. *IACR Cryptology ePrint Archive*, 2013:31, 2013.

[66] BSI. Kryptographische Verfahren: Empfehlungen und Schlüssellängen. BSI TR-02102 Version 2013.2, `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102_pdf.pdf?__blob=publicationFile`, 2013.

[67] Bundesnetzagentur. Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung. `http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/2013Algorithmenkatalog.pdf?__blob=publicationFile&v=1`, 2013.

[68] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[69] Christophe De Cannière and Christian Rechberger. Preimages for reduced SHA-0 and SHA-1. In Wagner [339], pages 179–202.

[70] Anne Canteaut and Kapalee Viswanathan, editors. *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*. Springer, 2004.

[71] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer, 2003.

[72] Certicom. Certicom announces elliptic curve cryptosystem (ECC) challenge winner. Certicom Press Release, 2009.

[73] Liqun Chen and Zhaohui Cheng. Security proof of Sakai-Kasahara's identity-based encryption scheme. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 442–459. Springer, 2005.

[74] Liqun Chen, Zhaohui Cheng, John Malone-Lee, and Nigel P. Smart. An efficient ID-KEM based on the Sakai–Kasahara key construction. *IEE Proc. Information Security*, 153:19–26, 2006.

[75] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Vaudenay [336], pages 1–11.

[76] Carlos Cid and Gaëtan Leurent. An analysis of the XSL algorithm. In Roy [301], pages 333–352.

[77] Don Coppersmith. Finding a small root of a bivariate integer equation; Factoring with high bits known. In Maurer [231], pages 178–189.

[78] Don Coppersmith. Finding a small root of a univariate modular equation. In Maurer [231], pages 155–165.

[79] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[80] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, and Michael K. Reiter. Low-exponent RSA with related messages. In Maurer [231], pages 1–9.

[81] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.

[82] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Knudsen [203], pages 272–287.

[83] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New attacks on PKCS#1 v1.5 encryption. In Preneel [286], pages 369–381.

[84] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On the security of RSA padding. In Wiener [344], pages 1–18.

[85] Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi, and Ralf-Philipp Weinmann. Practical cryptanalysis of ISO/IEC 9796-2 and EMV signatures. In Halevi [140], pages 428–444.

[86] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[87] F. Cusack and M. Forssen. Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). RFC 4256 (Proposed Standard), January 2006.

[88] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[89] W. Dai. An attack against SSH2 protocol. E-mail to the SECSH Working Group available from `ftp://ftp.ietf.org/ietf-mail-archive/secsh/2002-02.mail`, 6th Feb. 2002.

[90] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. On the joint security of encryption and signature in EMV. In Orr Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 116–135. Springer, 2012.

[91] Jean Paul Degabriele and Kenneth G. Paterson. Attacking the IPsec standards in encryption-only configurations. In *IEEE Symposium on Security and Privacy*, pages 335–349. IEEE Computer Society, 2007.

[92] Jean Paul Degabriele and Kenneth G. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 493–504. ACM, 2010.

[93] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176.

[94] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746, 6176.

[95] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[96] Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. Cryptology ePrint Archive, Report 2012/672, 2012. `http://eprint.iacr.org/`.

[97] Saar Drimer, Steven J. Murdoch, and Ross J. Anderson. Optimised to fail: Card readers for online banking. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 184–200. Springer, 2009.

[98] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In Rabin [290], pages 393–410.

[99] T. Duong and J. Rizzo. Here come the ⊕ ninjas. Unpublished manuscript, 2011.

[100] Thai Duong and Juliano Rizzo. The CRIME attack. Presentation at ekoparty Security Conference, 2012.

[101] Cynthia Dwork, editor. *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.

[102] E.A.Grechnikov. Collisions for 72-step and 73-step SHA-1: Improvements in the method of characteristics. Cryptology ePrint Archive, Report 2010/413, 2010. `http://eprint.iacr.org/`.

[103] D. Eastlake. Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH). RFC 4305 (Proposed Standard), December 2005. Obsoleted by RFC 4835.

[104] ECRYPT II NoE. ECRYPT II Yearly Report on Algorithms and Key Lengths (2008-2009). ECRYPT II deliverable D.SPA.7-1.0, 2009.

[105] ECRYPT II NoE. ECRYPT II Yearly Report on Algorithms and Key Lengths (2009-2010). ECRYPT II deliverable D.SPA.13-1.0, 2010.

[106] ECRYPT II NoE. ECRYPT II Yearly Report on Algorithms and Key Lengths (2010-2011). ECRYPT II deliverable D.SPA.17-1.0, 2011.

[107] ECRYPT II NoE. ECRYPT II Yearly Report on Algorithms and Key Lengths (2011-2012). ECRYPT II deliverable D.SPA.20-1.0, 2012.

[108] ECRYPT NoE. ECRYPT Yearly Report on Algorithms and Key Lengths (2004). ECRYPT deliverable D.SPA.10-1.1, 2004.

[109] ECRYPT NoE. ECRYPT Yearly Report on Algorithms and Key Lengths (2005). ECRYPT deliverable D.SPA.16-1.0, 2005.

[110] ECRYPT NoE. ECRYPT Yearly Report on Algorithms and Key Lengths (2006). ECRYPT deliverable D.SPA.21-1.0, 2006.

[111] ECRYPT NoE. ECRYPT Yearly Report on Algorithms and Key Lengths (2007-2008). ECRYPT deliverable D.SPA.28-1.0, 2008.

[112] EMV Co. Book 2 – Security and key management. EMV 4.3, 2011.

[113] ENISA. The use of cryptographic techniques in europe. http://www.enisa.europa.eu/activities/identity-and-trust/library/the-use-of-cryptographic-techniques-in-europe, 2011.

[114] ENISA. Recommended cryptographic measures - Securing personal data. https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/recommended-cryptographic-measures-securing-personal-data, 2013.

[115] ENISA. Work Programme 2013. http://www.enisa.europa.eu/publications/programmes-reports, 2013.

[116] Matthias Ernst, Ellen Jochemsz, Alexander May, and Benne de Weger. Partial key exposure attacks on RSA up to full size exponents. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 371–386. Springer, 2005.

[117] ETSI TS 102 176-. Electronic signatures and infrastructures (ESI); Algorithms and parameters for secure electronic signatures; Part 1: Hash functions and asymmetric algorithms. European Telecommunications Standards Institute, 2007.

[118] ETSI/SAGE Specification. Specification of the 3GPP Confidentiality and Integrity Algorithms. Document 2: Kasumi Algorithm Specification. ETSI/SAGE, 2011.

[119] EU. EC regulation (EU) No 611/2013 on the measures applicable to the notification of personal data breaches under Directive 2002/58/EC on privacy and electronic communications. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2013:173:0002:0008:en:PDF.

[120] Federal Information Processing Standards Publication 197. Specification for the advanced encryption standard (AES). National Institute of Standards and Technology, 2001.

[121] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of IPsec. Unpublished manuscript available from http://www.schneier.com/paper-ipsec.html, February 1999.

[122] Scott R. Fluhrer and Stefan Lucks. Analysis of the E_0 encryption system. In Vaudenay and Youssef [338], pages 38–48.

[123] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In Vaudenay and Youssef [338], pages 1–24.

[124] Jens Franke. RSA576. Post to various internet discussion boards/email lists, 2003.

[125] Jens Franke. RSA576. Post to various internet discussion boards/email lists, 2005.

[126] S. Frankel, R. Glenn, and S. Kelly. The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC 3602 (Proposed Standard), September 2003.

[127] S. Frankel and H. Herbert. The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec. RFC 3566 (Proposed Standard), September 2003.

[128] M. Friedl, N. Provos, and W. Simpson. Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4419 (Proposed Standard), March 2006.

[129] D. Fu and J. Solinas. Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2. RFC 5903 (Informational), June 2010.

[130] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Kilian [198], pages 260–274.

[131] M Peeters G. Bertoni, J. Daemen and G. Van Assche. The Keccak sponge function family. `http://keccak.noekeon.org/`.

[132] Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1):19–46, 2002.

[133] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[134] Danilo Gligoroski, Suzana Andova, and Svein J. Knapskog. On the importance of the key separation principle for different modes of operation. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *ISPEC*, volume 4991 of *Lecture Notes in Computer Science*, pages 404–418. Springer, 2008.

[135] Daniel M. Gordon. Discrete logarithms in GF(P) using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[136] GOST R 34-10-2001. Information technology – Cryptography data security – Formation and verification process of [electronic] signatures. State Standard of the Russion Federation, 2001.

[137] Robert Granger. Discrete logarithms in GF(2^{6120}). Post to NM-BRTHRY@LISTSERV.NODAK.EDU, 2013.

[138] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 56–75. Springer, 2010.

[139] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Canteaut and Viswanathan [70], pages 315–327.

[140] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*. Springer, 2009.

[141] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Okamoto [269], pages 292–304.

[142] Finn Michael Halvorsen, Olav Haugen, Martin Eian, and Stig Fr. Mjølsnes. An improved attack on TKIP. In Audun Jøsang, Torleiv Maseng, and Svein J. Knapskog, editors, *NordSec*, volume 5838 of *Lecture Notes in Computer Science*, pages 120–132. Springer, 2009.

[143] Helena Handschuh and Bart Preneel. Key-recovery attacks on universal hash function based MAC algorithms. In Wagner [339], pages 144–161.

[144] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard), November 1998. Obsoleted by RFC 4306, updated by RFC 4109.

[145] B. Harris. RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4432 (Proposed Standard), March 2006.

[146] Johan Håstad. Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, 17(2):336–341, 1988.

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

[148] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J.Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium – 2012*, pages 205–220, 2012.

[149] Miia Hermelin and Kaisa Nyberg. Correlation properties of the Bluetooth combiner generator. In JooSeok Song, editor, *ICISC*, volume 1787 of *Lecture Notes in Computer Science*, pages 17–29. Springer, 1999.

[150] Mathias Herrmann and Alexander May. Maximizing small root bounds by linearization and applications to small secret exponent RSA. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 53–69. Springer, 2010.

[151] Erwin Hess, Marcus Schafheutle, and Pascale Serf. The digital signature scheme ECGDSA, 2006.

[152] P. Hoffman. Cryptographic Suites for IPsec. RFC 4308 (Proposed Standard), December 2005.

[153] R. Housley. Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP). RFC 3686 (Proposed Standard), January 2004.

[154] R. Housley. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309 (Proposed Standard), December 2005.

[155] Nick Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptography*, 23(3):283–290, 2001.

[156] G. Hudson. Camellia Encryption for Kerberos 5. RFC 6803 (Informational), November 2012.

[157] IEEE 802.11. Wireless LAN medium access control MAC and physical layer PHY specifications. Institute of Electrical and Electronics Engineers Standard, 1999.

[158] IEEE 802.11-2012 (Revision of IEEE 802.11-2007). Wireless LAN medium access control MAC and physical layer PHY specifications. Institute of Electrical and Electronics Engineers Standard, 2012.

[159] IEEE 802.15.4 . Law rate WPAN. Institute of Electrical and Electronics Engineers Standard, 2012.

[160] IEEE P1363.3 (Draft D5). Identity-based public key cryptography using pairings. Institute of Electrical and Electronics Engineers Standard, 2012.

[161] K. Igoe and J. Solinas. AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. RFC 5647 (Informational), August 2009.

[162] Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other non-random properties for step-reduced SHA-256. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2008.

[163] Takanori Isobe, Yuhei Watanabe Toshihiro Ohigashi, and Masakatu Morii. Full plaintext recovery attack on broadcast RC4. In Moriai [245]. To appear.

[164] ISO/IEC 14888-3. Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms. International Organization for Standardization, 2009.

[165] ISO/IEC 14888-3. Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms – Ammendment 1. International Organization for Standardization, 2009.

[166] ISO/IEC 18033-2. Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric Ciphers. International Organization for Standardization, 2006.

[167] ISO/IEC 18033-4. Information technology – Security techniques – Encryption algorithms – Part 4: Stream ciphers. International Organization for Standardization, 2011.

[168] ISO/IEC 19972. Information technology – Security techniques – Authenticated encryption. International Organization for Standardization, 2009.

[169] ISO/IEC 29192-3. Information technology – Security techniques – Lightweight cryptography – Part 3: Stream ciphers. International Organization for Standardization, 2012.

[170] ISO/IEC 9796-2. Information technology – Security techniques – Digital signatures giving message recovery – Part 2: Integer factorization based schemes. International Organization for Standardization, 2010.

[171] ISO/IEC 9797-1:2011. Information technology – Security techniques – Digital signatures giving message recovery – Part 1: Mechanisms using a block cipher. International Organization for Standardization, 2011.

[172] ISO/IEC 9797-2:2011. Information technology – Security techniques – Digital signatures giving message recovery – Part 2: Mechanisms using a dedicated hash-function. International Organization for Standardization, 2011.

[173] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.

[174] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In Safavi-Naini and Canetti [304], pages 31–49.

[175] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. *IACR Cryptology ePrint Archive*, 2011:219, 2011.

[176] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Safavi-Naini and Canetti [304], pages 273–293.

[177] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013.

[178] Jakob Jonsson. Security proofs for the RSA-PSS signature scheme and its variants. Cryptology ePrint Archive, Report 2001/053, 2001. `http://eprint.iacr.org/`.

[179] Jakob Jonsson. On the security of CTR + CBC-MAC. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 76–93. Springer, 2002.

[180] Antoine Joux. Comments on the choice between CWC or GCM – authentication weaknesses in GCM. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf`.

[181] Antoine Joux. Comments on the draft GCM specification – authentication failures in NIST version of GCM. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf`.

[182] Antoine Joux. Discrete logarithms in GF(2^{6168}). Post to NM-BRTHRY@LISTSERV.NODAK.EDU, 2013.

[183] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In Johansson and Nguyen [177], pages 177–193.

[184] Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. *IACR Cryptology ePrint Archive*, 2013:95, 2013.

[185] Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In Dwork [101], pages 326–344.

[186] Hendrik W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3):649–673, 1987.

[187] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 537–553. Springer, 2012.

[188] Ju-Sung Kang, Sang Uk Shin, Dowon Hong, and Okyeon Yi. Provable security of KASUMI and 3GPP encryption mode f8. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2001.

[189] Orhun Kara and Cevat Manap. A new class of weak keys for Blowfish. In Biryukov [43], pages 167–180.

[190] A. Kato, M. Kanda, and S. Kanno. Modes of Operation for Camellia for Use with IPsec. RFC 5529 (Proposed Standard), April 2009.

[191] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Obsoleted by RFC 5996, updated by RFC 5282.

[192] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), September 2010. Updated by RFC 5998.

[193] S. Kelly and S. Frankel. Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec. RFC 4868 (Proposed Standard), May 2007.

[194] S. Kent. Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP). RFC 4304 (Proposed Standard), December 2005.

[195] S. Kent. IP Authentication Header. RFC 4302 (Proposed Standard), December 2005.

[196] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), December 2005.

[197] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005. Updated by RFC 6040.

[198] Joe Kilian, editor. *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.

[199] Aleksandar Kircanski and Amr M. Youssef. On the sliding property of SNOW 3G and SNOW 2.0. *IET Information Security*, 5(4):199–206, 2011.

[200] T. Kivinen and M. Kojo. More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). RFC 3526 (Proposed Standard), May 2003.

[201] Thorsten Kleinjung. Discrete logarithms in GF(p) — 160 digits. Post to NMBRTHRY@LISTSERV.NODAK.EDU, 2007.

[202] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman J. J. te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. In Rabin [290], pages 333–350.

[203] Lars R. Knudsen, editor. *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002.

[204] John T. Kohl. The use of encryption in Kerberos for network authentication. In Brassard [62], pages 35–43.

[205] Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A high-performance conventional authenticated encryption mode. In Roy and Meier [302], pages 408–426.

[206] Geir M. Køien and Vladimir A. Oleshchuk. Location privacy for cellular systems; analysis and solution. In George Danezis and David Martin, editors, *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 40–58. Springer, 2005.

[207] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Best Current Practice), February 1997.

[208] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In *CRYPTO*, 2013.

[209] T. Krovetz. UMAC: Message Authentication Code using Universal Hashing. RFC 4418 (Best Current Practice), March 2006.

[210] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In Matsui [230], pages 126–143.

[211] Franck Landelle and Thomas Peyrin. Cryptanalysis of full RIPEMD-128. In Johansson and Nguyen [177], pages 228–244.

[212] Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.

[213] Dong Hoon Lee and Xiaoyun Wang, editors. *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*. Springer, 2011.

[214] Arjen Lenstra. Key lengths. In Hossein Bidgoli, editor, *Handbook of Information Security: Volume II: Information Warfare; Social Legal, and International Issues; and Security Foundations*, pages 617–635. Wiley, 2004.

[215] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, Whit is right. *IACR Cryptology ePrint Archive*, 2012:64, 2012.

[216] Arjen K. Lenstra and Hendrik W. Lenstra. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.

[217] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Datenschutz und Datensicherheit*, 24(3), 2000.

[218] M. Lepinski and S. Kent. Additional Diffie-Hellman Groups for Use with IETF Standards. RFC 5114 (Informational), January 2008.

[219] Gaëtan Leurent. Message freedom in MD4 and MD5 collisions: Application to APOP. In Biryukov [43], pages 309–328.

[220] Chu-Wee Lim and Khoongming Khoo. An analysis of XSL applied to BES. In Biryukov [43], pages 242–253.

[221] J. Linn. Generic Security Service Application Program Interface Version 2, Update 1. RFC 2743 (Proposed Standard), January 2000. Updated by RFC 5554.

[222] Moses Liskov and Kazuhiko Minematsu. Comments on the proposal to approve XTS-AES – Comments on XTS-AES. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Liskov_Minematsu.pdf`.

[223] Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In Shoup [320], pages 97–117.

[224] C. Madson and R. Glenn. The Use of HMAC-MD5-96 within ESP and AH. RFC 2403 (Proposed Standard), November 1998.

[225] C. Madson and R. Glenn. The Use of HMAC-SHA-1-96 within ESP and AH. RFC 2404 (Proposed Standard), November 1998.

[226] Subhamoy Maitra and Goutam Paul. New form of permutation bias and secret key leakage in keystream bytes of RC4. In Nyberg [267], pages 253–269.

[227] James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0. In Kilian [198], pages 230–238.

[228] M. Matsui, J. Nakajima, and S. Moriai. A Description of the Camellia Encryption Algorithm. RFC 3713 (Informational), April 2004.

[229] Mitsuru Matsui. Linear cryptoanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[230] Mitsuru Matsui, editor. *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*. Springer, 2009.

[231] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[232] Alexander Maximov and Alex Biryukov. Two trivial attacks on Trivium. In Adams et al. [6], pages 36–55.

[233] Alexander Maximov and Dmitry Khovratovich. New state recovery attack on RC4. In Wagner [339], pages 297–316.

[234] D. McGrew and D. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655 (Best Current Practice), July 2012.

[235] D. McGrew and J. Viega. The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. RFC 4543 (Proposed Standard), May 2006.

[236] David A. McGrew. Efficient authentication of large, dynamic data sets using Galois/Counter mode (GCM). In *IEEE Security in Storage Workshop*, pages 89–94. IEEE Computer Society, 2005.

[237] David A. McGrew and John Viega. The security and performance of the Galois/Counter mode (GCM) of operation. In Canteaut and Viswanathan [70], pages 343–355.

[238] Florian Mendel, Tomislav Nad, Stefan Scherz, and Martin Schläffer. Differential attacks on reduced ripemd-160. In Dieter Gollmann and Felix C. Freiling, editors, *ISC*, volume 7483 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2012.

[239] Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced SHA-256. In Johansson and Nguyen [177], pages 262–278.

[240] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. On the collision resistance of RIPEMD-160. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC*, volume 4176 of *Lecture Notes in Computer Science*, pages 101–116. Springer, 2006.

[241] Florian Mendel, Christian Rechberger, and Martin Schläffer. Update on SHA-1. Presented at Rump Session of Crypto 2007, 2007.

[242] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[243] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. In *In Project Athena Technical Plan*, 1987.

[244] Vebjørn Moen, Håvard Raddum, and Kjell Jørgen Hole. Weaknesses in the temporal key hash of WPA. *Mobile Computing and Communications Review*, 8(2):76–83, 2004.

[245] Shiho Moriai, editor. *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2013.

[246] Masakatu Morii and Yosuke Todo. Cryptanalysis for RC4 and breaking WEP/WPA-TKIP. *IEICE Transactions*, 94-D(11):2087–2094, 2011.

[247] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The TLS handshake protocol: A modular analysis. *J. Cryptology*, 23(2):187–223, 2010.

[248] Steven J. Murdoch, Saar Drimer, Ross J. Anderson, and Mike Bond. Chip and pin is broken. In *IEEE Symposium on Security and Privacy*, pages 433–446. IEEE Computer Society, 2010.

[249] Sean Murphy and Matthew J. B. Robshaw. Essential algebraic structure within the AES. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

[250] Mridul Nandi. A unified method for improving PRF bounds for a class of blockcipher based MACs. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 212–229. Springer, 2010.

[251] National Security Agency. Suite b cryptography. `http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`, 2009.

[252] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[253] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649, 6806.

[254] Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for Schnorr signatures. *J. Mathematical Cryptology*, 3(1):69–87, 2009.

[255] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002.

[256] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Des. Codes Cryptography*, 30(2):201–217, 2003.

[257] NIST Special Publication 800-108. Recommendation for key derivation using pseudorandom functions. National Institute of Standards and Technology, 2009.

[258] NIST Special Publication 800-38A. Recommendation for block cipher modes of operation – Modes and techniques. National Institute of Standards and Technology, 2001.

[259] NIST Special Publication 800-38C. Recommendation for block cipher modes of operation – The CCM mode for authentication and confidentiality. National Institute of Standards and Technology, 2004.

[260] NIST Special Publication 800-38D. Recommendation for block cipher modes of operation – Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology, 2007.

[261] NIST Special Publication 800-38E. Recommendation for block cipher modes of operation – The XTS-AES mode for confidentiality on storage devices. National Institute of Standards and Technology, 2010.

[262] NIST Special Publication 800-56A. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. National Institute of Standards and Technology, 2007.

[263] NIST Special Publication 800-56B. Recommendation for pair-wise key establishment schemes using integer factorization cryptography. National Institute of Standards and Technology, 2009.

[264] NIST Special Publication 800-56C. Recommendation for key derivation through extraction-then-expansion. National Institute of Standards and Technology, 2009.

[265] NIST Special Publication 800-57. Recommendation for key management – Part 1: General (Revision 3). National Institute of Standards and Technology, 2012.

[266] NIST Special Publication 800-67-Rev1. Recommendation for the triple data encryption standard algorithm (tdea) block cipher. National Institute of Standards and Technology, 2012.

[267] Kaisa Nyberg, editor. *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*. Springer, 2008.

[268] Kaisa Nyberg and Johan Wallén. Improved linear distinguishers for SNOW 2.0. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 144–162. Springer, 2006.

[269] Tatsuaki Okamoto, editor. *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*. Springer, 2004.

[270] H. Orman and P. Hoffman. Determining Strengths For Public Keys Used For Exchanging Symmetric Keys. RFC 3766 (Best Current Practice), April 2004.

[271] Kenneth G. Paterson. A cryptographic tour of the IPsec standards. Cryptology ePrint Archive, Report 2006/097, 2006. `http://eprint.iacr.org/`.

[272] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn,*

Estonia, May 15-19, 2011. Proceedings, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.

[273] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the tls record protocol. In Lee and Wang [213], pages 372–389.

[274] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In Lee and Wang [213], pages 161–178.

[275] Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In Gilbert [133], pages 345–361.

[276] Kenneth G. Paterson and Arnold K. L. Yau. Padding Oracle Attacks on the ISO CBC Mode Encryption Standard. In Okamoto [269], pages 305–323.

[277] Kenneth G. Paterson and Arnold K. L. Yau. Cryptography in theory and practice: The case of encryption in IPsec. In Vaudenay [336], pages 12–29.

[278] R. Pereira and R. Adams. The ESP CBC-Mode Cipher Algorithms. RFC 2451 (Proposed Standard), November 1998.

[279] Erez Petrank and Charles Rackoff. CBC MAC for real-time data sources. *J. Cryptology*, 13(3):315–338, 2000.

[280] Krzysztof Pietrzak. A tight bound for EMAC. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2006.

[281] PKCS #1 v1.5. RSA cryptography standard. RSA Laboratories, 1993.

[282] PKCS #1 v2.1. RSA cryptography standard. RSA Laboratories, 2002.

[283] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[284] David Pointcheval and Serge Vaudenay. On provable security for digital signature algorithms. Technical Report LIENS-96-17, 1996.

[285] John M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comput.*, 32(143):918–924, 1978.

[286] Bart Preneel, editor. *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000.

[287] Bart Preneel and Paul C. van Oorschot. MDx-MAC and building fast MACs from hash functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1995.

[288] Bart Preneel and Paul C. van Oorschot. On the security of iterated message authentication codes. *IEEE Transactions on Information Theory*, 45(1):188–199, 1999.

[289] Gorden Proctor and Carlos Cid. On weak keys and forgery attacks against polynomial-based MAC schemes. In Moriai [245]. To appear.

[290] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[291] K. Raeburn. Advanced Encryption Standard (AES) Encryption for Kerberos 5. RFC 3962 (Proposed Standard), February 2005.

[292] K. Raeburn. Encryption and Checksum Specifications for Kerberos 5. RFC 3961 (Proposed Standard), February 2005.

[293] Vincent Rijmen. *Cryptanalysis and design of iterated block ciphers*. PhD thesis, Katholieke Universiteit Leuven, 1997.

[294] Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS*. The Internet Society, 2010.

[295] P. Rogaway. Problems with proposed IP cryptography. Available at `http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt`, 61995.

[296] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[297] Phillip Rogaway. Evaluation of some blockcipher modes of operation. Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan, 2011.

[298] Phillip Rogaway. Free OCB licenses. `http://www.cs.ucdavis.edu/~rogaway/ocb/license.htm`, 2013.

[299] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[300] Phillip Rogaway and David Wagner. A critique of CCM. Cryptology ePrint Archive, Report 2003/070, 2003. `http://eprint.iacr.org/`.

[301] Bimal K. Roy, editor. *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*. Springer, 2005.

[302] Bimal K. Roy and Willi Meier, editors. *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*. Springer, 2004.

[303] Markku-Juhani Olavi Saarinen. Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.

[304] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012.

[305] J. Salowey, A. Choudhury, and D. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288 (Proposed Standard), August 2008.

[306] Somitra Kumar Sanadhya and Palash Sarkar. New collision attacks against up to 24-step SHA-2. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 91–103. Springer, 2008.

[307] Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 378–396. Springer, 2011.

[308] Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2009.

[309] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. Security of MD5 challenge and response: Extension of APOP password recovery attack. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[310] Takakazu Satoh and Kiyomichi Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Math. Univ. St. Pauli*, 47:81–92, 1998.

[311] J. Schiller. Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2). RFC 4307 (Proposed Standard), December 2005.

[312] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In Ross J. Anderson, editor, *FSE*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 1993.

[313] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Brassard [62], pages 239–252.

[314] SEC 1. Elliptic curve cryptography – version 2.0. Standards for Efficient Cryptography Group, 2009.

[315] SEC 2. Recommended elliptic curve domain parameters – version 2.0. Standards for Efficient Cryptography Group, 2010.

[316] Igor A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comput.*, 67(221):353–356, 1998.

[317] Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux. Statistical attack on RC4 - distinguishing WPA. In Paterson [272], pages 343–363.

[318] Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux. Statistical attack on rc4 - distinguishing wpa. In Paterson [272], pages 343–363.

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

[320] Victor Shoup, editor. *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*. Springer, 2005.

[321] Nigel P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.

[322] Nigel P. Smart. Errors matter: Breaking rsa-based pin encryption with thirty ciphertext validity queries. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 15–25. Springer, 2010.

[323] D. Stebila and J. Green. Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer. RFC 5656 (Proposed Standard), December 2009.

[324] Marc Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In Johansson and Nguyen [177], pages 245–261.

[325] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.

[326] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and applications. *IJACT*, 2(4):322–359, 2012.

[327] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen K. Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In Halevi [140], pages 55–69.

[328] Erik Tews and Martin Beck. Practical attacks against wep and wpa. In David A. Basin, Srdjan Capkun, and Wenke Lee, editors, *WISEC*, pages 79–86. ACM, 2009.

[329] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2007.

[330] Yosuke Todo, Yuki Ozawa, Toshihiro Ohigashi, and Masakatu Morii. Falsification attacks against WPA-TKIP in a realistic environment. *IEICE Transactions*, 95-D(2):588–595, 2012.

[331] TTA.KO-12.0001/R1. Digital signature scheme with appendix – Part 2: Certificate-based digital signature algorithm. Korean Telecommunications Technology Association, 2000.

[332] Kyushu University, NICT, and Fujitsu Laboratories. Achieve world record cryptanalysis of next-generation cryptography. `http://www.nict.go.jp/en/press/2012/06/PDF-att/20120618en.pdf`, 2012.

[333] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999.

[334] Serge Vaudenay. On the weak keys of Blowfish. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 27–32. Springer, 1996.

[335] Serge Vaudenay. Security flaws induced by CBC padding - Applications to SSL, IPSEC, WTLS ... In Knudsen [203], pages 534–546.

[336] Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[337] Serge Vaudenay and Martin Vuagnoux. Passive-only key recovery attacks on RC4. In Adams et al. [6], pages 344–359.

[338] Serge Vaudenay and Amr M. Youssef, editors. *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*. Springer, 2001.

[339] David Wagner, editor. *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*. Springer, 2008.

[340] Xiaoyun Wang. New collision search for SHA-1. Presented at Rump Session of Crypto 2005, 2005.

[341] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Shoup [320], pages 17–36.

[342] Doug Whiting, Russ Housley, and Neils Ferguson. Submission to NIST: Counter with CBC-MAC (CCM) – AES mode of operation. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/ccm.pdf`.

[343] Michael J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[344] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[345] Stephen C. Williams. Analysis of the SSH key exchange protocol. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2011.

[346] Arnold K. L. Yau, Kenneth G. Paterson, and Chris J. Mitchell. Padding oracle attacks on CBC-mode encryption with secret and random IVs. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 299–319. Springer, 2005.

[347] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol. RFC 4252 (Proposed Standard), January 2006.

[348] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), January 2006.

[349] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Proposed Standard), January 2006. Updated by RFC 6668.

[350] L. Zhu and B. Tung. Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). RFC 4556 (Proposed Standard), June 2006. Updated by RFC 6112.

Index