Algorithms, key size and parameters report – 2014

November, 2014

European Union Agency for Network and Information Security www.enisa.europa.eu

About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Authors

Contributors to this report:

This work was commissioned by ENISA under contract ENISA D-COD-14-TO9 (under F-COD-13-C23) to the consortium formed by K.U.Leuven (BE) and University of Bristol (UK).

- Contributors: Nigel P. Smart (University of Bristol), Vincent Rijmen (KU Leuven), Benedikt Gierlichs (KU Leuven), Kenneth G. Paterson (Royal Holloway, University of London), Martijn Stam (University of Bristol), 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: Michel Abdalla (ENS Paris), Kenneth G. Paterson (Royal Holloway, University of London), Ahmad-Reza Sadeghi (T.U. Darmstadt), Michael Ward (Mastercard) for their comments suggestions and feedback.
- We also thank a number of people for providing anonymous input and Cas Cremers (Oxford University) and Hugo Krawczyk (IBM) for detailed comments on various aspects.

Contact

For contacting the authors please use sta@enisa.europa.eu.

For media enquires about this paper, please use press@enisa.europa.eu.

Legal notice

Copyright Notice

Catalogue number TP-05-14-084-EN-N ISBN 978-92-9204-102-1 DOI 10.2824/36822

Contents

Acronyms

3DES	Triple DES
3GPP	3rd Generation Partnership Project (mobile phone system)
A5/X	Stream ciphers used in mobile phone protocols
ABE	Attribute Based Encryption
AE	Authenticated Encryption
AEAD	Authenticated Encryption with Auxillary Data
AES	Advanced Encryption Standard
AESKW	A Key Wrap Scheme
AH	Authentication Header
AKA	Authentication and Key Agreement
AKW1	A Key Wrap Scheme
AKW2	A Key Wrap Scheme
AMAC	ANSI Retail MAC
ANSI	American National Standards Institute
BB	Boneh–Boyen (ID based encryption)
BF	Boneh–Franklin (ID based encryption)
BPP	Binary Packet Protocol
BPR	Bellare, Pointcheval and Rogaway
BMP	Boyko, MacKenzie and Patel
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

DAA	Direct Anonymous Attestation
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)
EC2	Elastic Computing Cloud
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
EKE	Encrypted Key Exchange
EMAC	Encrypted CBC-MAC
EME	ECB-mask-ECB (mode)
EMV	Europay–Mastercard–Visa (chip-and-pin system)
ENISA	European Union Agency for Network and Information Security
ESP	Encapsulating Security Payload
FDH	Full Domain Hash
FHE	Fully Homomorphic Encryption
GCM	Galois Counter Mode
GDSA	German Digital Signature Algorithm
GMAC	The MAC part of the GCM block cipher mode
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)
J-PAKE	Password Authenticated Key Exchange by Juggling
KDSA	Korean Digital Signature Algorithm
KDF	Key Derivation Function
KEM	Key Encapsulation Mechanism
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
LAMP	Linux, Apache, MySQL, PHP
LTE	Long Term Evolution (mobile phone system)
LWE	Learning With Errors
MAC	Message Authentication Code
MOV	Menezes–Okamoto–Vanstone (attack)
MPC	Multi Party Computation
MQV	Menezes–Qu–Vanstone (protocol)
MS	Mobile Station (i.e. a phone in UMTS/LTE)
NIST	National Institute of Standards and Technology (US)
NMAC	Nested MAC
NTRU	A Post Quantum Encryption Algorithm

OAEP	Optimal Asymmetric Encryption Padding
OCB	Offset Code Book (mode)
OFB	Output Feedback (mode)
OPE	Order Preserving Encryption
ORAM	Oblivious Random Access Memory
PACE	Password Authenticated Connection Establishment
PAK	Password Authenticated Key (PAK) Exchange
PAKE	Password Authenticated Key Exchange
PEKS	Public Key Encryption Keyword Search
PKCS	Public Key Cryptography Standards
PKE	Public Key Encryption
POR	Proofs Of Retrievability
PRF	Pseudo Random Function
PRNG	Pseudo Random Number Generator
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
RFC	Request For Comments
RAM	Random Access Memory
ROM	Random Oracle Model
RSA	Rivest–Shamir–Adleman

SA	Security Association
SHA	Secure Hash Algorithm
SIMD	Single Instruction Multiple Data
SIV	Synthetic Initialization Vector
SK	Sakai–Kasahara (ID-based encryption)
SN	Serving Network (i.e. a provider in UMTS/LTE)
SPAKE	Single-Party Public-Key Authenticated Key Exchange
SPD	Security Policy Database
SQL	Structured Query Language
SSE	Symmetric Searchable Encryption
SSH	Secure Shell
SSL	Secure Sockets Layer
TKW	Triple-DEA Key Wrap
TDKW	A Key Wrap Scheme
TRNG	True Random Number Generator
UEA	UMTS Encryption Algorithm
UF	Universally Unforgeable
UIA	UMTS Integrity Algorithm
UMAC	Universal hashing based MAC
UMTS	Universal Mobile Telecommunications System
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing

Chapter 1

Executive Summary

During 2013, ENISA prepared and published its first reports with cryptographic guidelines supporting the security measures required to protect personal data in online systems. Recently published EC Regulations on the measures applicable to the notification of personal data breaches [118] make reference to ENISA, as a consultative body, in the process of establishing a list of appropriate cryptographic protective measures.

This report is providing an update of the 2013 report [113] produced by ENISA. As was the case with the report of 2013, the cryptographic guidelines of ENISA should serve as a reference document, and cannot fill in for the existing lack of cryptographic recommendations at EU level. As such we provide rather conservative guiding principles, based on current state-of-the-art research, addressing construction of new systems with a long life cycle. This report is aimed to be a reference in the area, focusing on commercial online services that collect, store and process the personal data of EU citizens.

In the report of 2013 there was a section on protocols; for this year we decided to extend the part on implementation by adding to this report a section on side-channels, random number generation, and key life cycle management. The summary of protocols is now covered in a sister report [114].

It should be noted that this is a technical document addressed to decision makers, in particular specialists designing and implementing cryptographic solutions, within commercial organisations. In this document we focus on just two decisions which we feel are more crucial to users of cryptography.

Firstly, whether a given primitive or scheme can be considered for use today if it is already deployed. We refer to such use as *legacy* use within our document. Our first guiding principle 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 advise that the primitive or scheme should be replaced as a matter of urgency.

Secondly, we consider the issue of whether a primitive or scheme is suitable for deployment in new or future systems. In some sense mechanisms which we consider usable 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 guiding principle 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.

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.

Further limitations of scope are mentioned in the introductory chapter which follows. Further restrictions are mentioned 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 proposals for algorithm and keysizes The 2013 version of this report [113] also contained a section on protocol proposals; as remarked in the executive summary this has now been separated into a separate report [114]. 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 proposals in an easy to use form. The prior ECRYPT documents provided more general background information and discussions on general concepts with respect to 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, 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 *strong advice* that the primitive or scheme be possibly *replaced* as a matter of urgency (or even that an attack exists). A system which we deem not secure for legacy use may still actually be secure if used within a specific environment, e.g. limited key life times, mitigating controls, or (in the case of hash functions) relying on non-collision resistance properties. However, in such instances we advise the user consults expert advise to see whether such specific details are indeed relevant.

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 schemes 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, 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

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

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 and standardized.

As a general rule of thumb we consider symmetric 80-bit security levels to be sufficient for legacy applications for the coming years[1], but consider 128-bit security levels to be the minimum requirement for new systems being deployed. Thus the key take home message 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 proposals in this document need to be read with this in mind. We concentrate on proposals which imply a minimal security level across all applications; i.e. the most conservative approach. Thus this does not imply that a specific application cannot use security levels lower than 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.

Unlike the previous report [113] we consider implementation issues such as side channels resulting from timing, power, cache analysis etc, insufficient randomness generation and key life-cycle management; as well as implementation issues related to the mathematical instantiation of the

[1] An exception is made for SHA-1: although it (probably) does not offer 80-bit security, it is still included for legacy use in this year's document. However, we propose removing SHA-1 from applications as soon as possible .

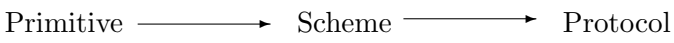
scheme, such as padding oracle attacks etc.

As a 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 Understanding Terminology and Structure

The document divides cryptographic mechanisms into primitives (such as block ciphers, public key primitive and hash functions) and schemes (such as symmetric and public key encryption schemes, signature schemes etc).

Protocols (such as key agreement, TLS, IPsec etc), discussed in [114], are themselves 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 in this report such proofs do exist.



Cryptographic primitives 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 cryptographic community's current "best guess". We discuss primitives in detail in Chapter 3.

In Chapter 4 we then go onto discuss basic cryptographic schemes, and in Chapter 5 we discuss more advanced or esoteric 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. Cryptographic protocols are dealt with in the companion report [114].

In this 2014 report we add in a new chapter, Chapter 6, on a number of general issues related to the deployment of cryptographic primitives and schemes. In this edition of the report we restrict ourselves to hardware and software side-channels, to random number generation and to key life-cycle management.

2.2 Making a Decision

Making the distinction between schemes and primitives 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. A security proof which reduces security of a scheme to the security of an underlying primitive can introduce a *security loss*. The "loss" is the proportion of additional effort an attacker who can break the scheme needs to expend so as to break the primitive. This loss leads some cryptographers to state that the key size of the primitive should be chosen with respect to this loss. With such a decision, unless proofs are *tight*[2], 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 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.

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

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

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 3072 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 Chapters 4 and 5 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 details on 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 choices are often rather conservative since we aim to give proposals for the constructions of systems with a long life cycle.

As already remarked this is an update of the 2013 ENISA report [113], which was itself strongly related to the ECRYPT and ECRYPT2 reports [104–111]. As mentioned earlier the current document is focused on making explicit proposals as opposed to providing a general framework and summary as the original ECRYPT documents did.

Various government organisations provide advice, see annex A of [112], 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 [16] published by France, and [63,64] published by Germany. The key size recommendations of these three documents are in almost all cases consistent with our own proposals for symmetric key sizes, hash function sizes and elliptic curve key sizes. The documents [63] and [16] also mention integer factorisation based primitives; our proposals are more conservative than these two documents. Along with [16] we place a strong emphasis on using schemes with modern security proofs.

Further afield the US government maintains a similar document called Suite B [243], which presents recommended algorithms and key sizes for various governmental uses. Again our analysis is broadly consistent in terms of key sizes with this document.

All of these documents [16, 63, 64, 243] 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 primitives, and some integer factorisation 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 systems designed a long time ago but which are still in use (e.g. MD5, SHA-1, DES etc).

In terms of cryptographic schemes our coverage is much wider than that of [63, 64, 243]; 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 [63, 64, 243] in that we propose the DSA algorithm, and many of its variants, for use in legacy systems only. This is because DSA only has a security proof in a relatively weak computational model [61]. For discrete logarithm based signatures we propose schemes such as Schnorr signatures [318], which have stronger provable security properties than DSA [244, 280].

Another form of comparison can be made with the documents of various standards organisations. The ones which have been most referred to in this report are those of IETF, ISO and NIST. Divergences from our analysis (if any) in these standards are again due to the distinct audiences. The IETF standardises the protocols which keeps the internet running, their main concern is hence interoperability. As we have seen in recent years, with attacks on TLS and IPSec, this often leads to compromises in algorithm selection and choice. The ISO takes a very liberal approach to standardising 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.

Finally, we compare with the recommendations of the European Payments Council (EPC). In their document [119] the EPC also divide cryptographic systems into those for legacy and those for future use. They classify SHA-1, RSA moduli with 1024 bits, ECC keys of 160 bits as suitable for legacy use, and 3DES, AES-128, SHA-2 (256 and 512 bit variants), SHA-3, Whirlpool, RSA moduli with 2048 bits, ECC keys of 224 bits or more as suitable for future use. These are broadly in line with our analysis.

2.4 Open Issues and Areas Not Covered

Much of the analysis in this document is 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 proposals 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: The reason for examining these now is that systems currently being deployed may need to be resistant against the future development of a quantum computer, or may need to be designed so that a switch to a post-quantum system is simple.
- Short signatures and signatures with message recovery: Short signatures are used in multiple scenarios, and signatures with message recovery are used in currently deployed systems such as the chip-and-pin system EMV. The current document does not cover such cryptographic schemes.
- Encryption schemes which enable de-duplication of ciphertexts: The use of such schemes, and other deterministic encryption schemes such as format preserving encryption, are becoming

more used in real systems. Encryption which enables de-duplication is important to enable secure cloud backup.

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.

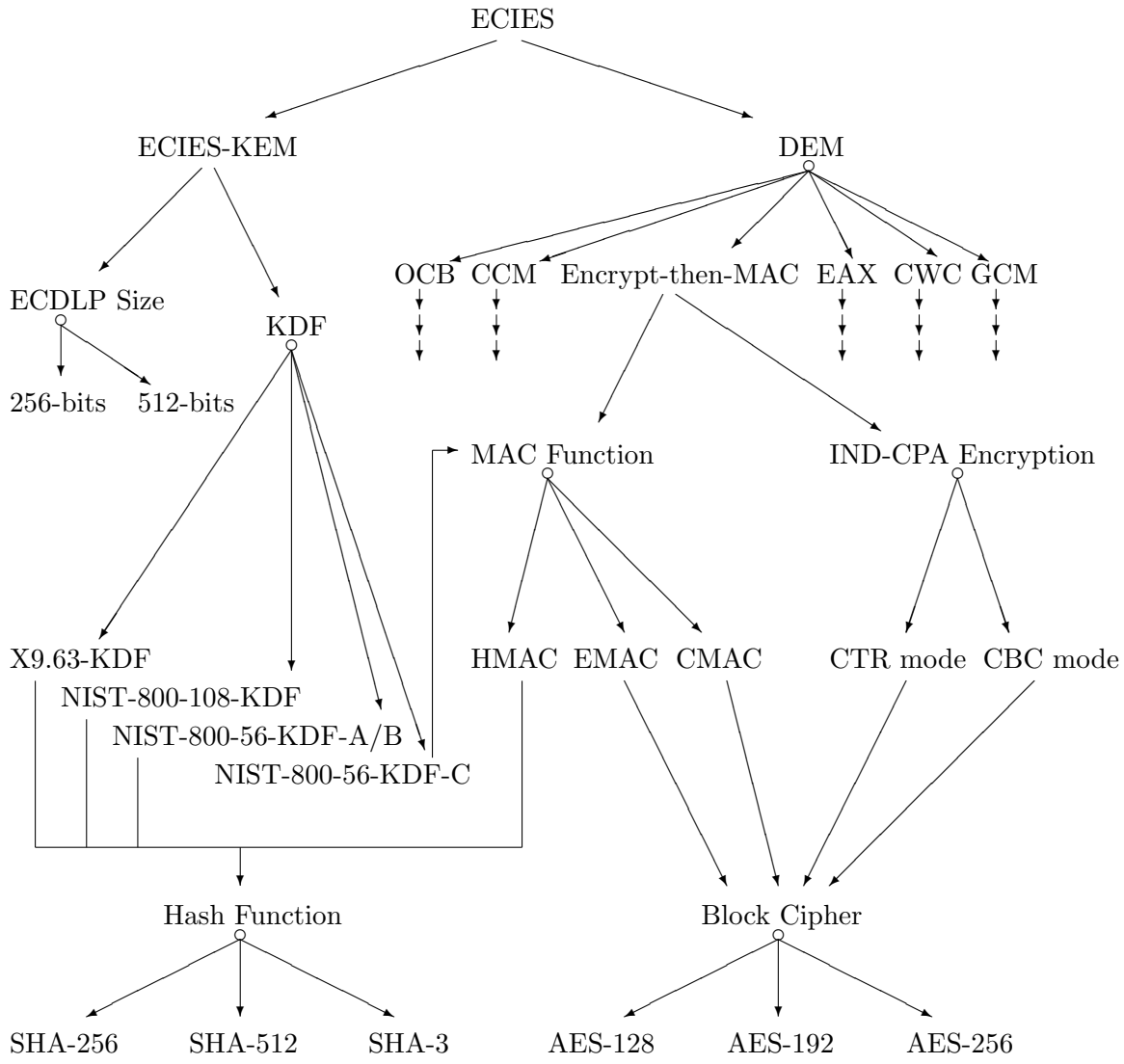


Figure 2.1: Just some of the design space for instantiating the ECIES public key encryption algorithm. Note, that not all standards documents will support all of these options. To read this diagram: A group of arrows starting with a circle implies the implementer needs to choose one of the resulting 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, e.g. some hash functions can be block cipher based. 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 primitives within the table in order of security strength (usually).

3.1 Comparison

In making a decision as to which cryptographic mechanism to employ, one first needs to choose the mechanism and then decide on 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 or the use of short lived session keys. 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] ⋆														
80	1184	142	112	3808	200	128	5888	230	192	20160	350	256	46752	474
Lenstra 2004 [214] ⋆														
80	1329	160	112	3154	224	128	4440	256	192	12548	384	256	26268	512
IETF 2004 [268] ⋆														
80	1233	148	112	2448	210	128	3253	242	192	7976	367	256	15489	494
SECG 2009 [319]														
80	1024	160	112	2048	224	128	3072	256	192	7680	384	256	15360	512
NIST 2012 [262]														
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 ⋆ 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 [259] 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 [285, 286], 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 3072-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.

Some cryptanalysists include the resistance against related-key attacks in the security evaluation of a block cipher. We include these results for completeness. Note however that the existence of a related-key attack on a given block cipher does *not* contradict the assumption that the block cipher acts as a pseudo-random permutation. Furthermore, the soundness of security models allowing for related-key attacks is still under investigation.

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. We also consider that the maximum amount of data which should be encrypted under the same key should be bounded by $2^{n/2}$ blocks, 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 advise 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 guidelines for minimum parameter sizes for future and legacy use. For convenience these guidelines are summarised in Table 3.2.

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

Table 3.2: Block Cipher Summary

3.2.1 Future Use Block Ciphers

AES

The Advanced Encryption Standard, or AES, is the block cipher of choice for future applications [89, 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 [87, 240]. However, such attacks have not been shown to be effective [77, 219].

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 [51] 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 [224]. 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 [263]. 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.

OBSERVATION: Due to meet-in-the-middle attacks 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 [339]. For the three key variant the security is reduced to 2^{112}.

OBSERVATION: For both variants, related-key attacks with complexity 2^{88} are published [275]. For the three-key variant, a trivial related-key attack for the related keys $k_1\|k_2\|k_3$ and $\overline{k_1}\|\overline{k_2}\|k_3$, where $\overline{k}$ is the bitwise complement of k, with complexity of 2^{56} exists in the folk lore.

Kasumi

This cipher [117], used in 3GPP, has a 128-bit key and 64-bit block size is a variant of MISTY-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 [100] 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 advise to use Kasumi in further applications.

Blowfish

This cipher [317] has a 64-bit block size, which is too small for some applications and the reason we only advise it for legacy use. It also has a key size ranging from 32- to 448-bits, which we clearly only endorse 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, 292, 341] but no attacks on the full cipher.

3.2.3 Historical (non-endorsed) 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 [225].

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 functions 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. Hash functions built from block ciphers, as considered in [159] are not considered in this report.

3.3.1 Future Use 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

Primitive	Output Length	Classification Legacy	Classification Future
SHA-2	256, 384, 512	✓	✓
SHA3	256,384,512	✓	✓
Whirlpool	512	✓	✓
SHA3	224	✓	✗
SHA-2	224	✓	✗
RIPEMD-160	160	✓	✗
SHA-1	160	✓[1]	✗
MD-5	128	✗	✗
RIPEMD-128	128	✗	✗

Table 3.3: Hash Function Summary

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 [156, 233, 310]. In addition reduced round variants 43 (resp. 46) have also been attacked for preimage resistance [17, 136].

SHA3

The competition organised by NIST to find an algorithm for SHA3 ended on October 2nd, 2012, with the selection of Keccak [127]. In April 2014, a draft version of FIPS 202 describing SHA3 has been released [121]. The draft standard contains 4 hash functions: SHA3-224, SHA3-256, SHA3-384 and SHA3-512.

OBSERVATION: Reduced round collision attacks (4 out of 24) have been reported [95]. For applications that use a secret key as part of the input of a hash function, cube attacks with practical complexity have been shown for up to 6 rounds of Keccak [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 [311], as well as collisions on 5.5 rounds [210], with complexity 2^{120}. In [314] this is extended to 6 rounds, with 2^{481} computation cost. Collision attacks are also given in [314] where eight rounds are attacked with complexity 2^{120}.

3.3.2 Legacy Hash Functions

RIPEMD-160

Collision attacks on 36 rounds (out of 80) have been found [232]. These were extended to 42 rounds in [234].

SHA-1

SHA-1 is in widespread use and was designed to provide protection against collision finding of 2^{80}, it was standardized in NIST-180-4 [248]. However, several authors claim that collisions can be found with a computational effort that is significantly lower [236, 346, 347]. The current best analysis is that of 2^{61} operations, reported in [330]. 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].

Due to the importance we repeat the footnote from Page 12: We have decided to keep SHA-1 as a legacy use algorithm since SHA-3 has not yet been officially standardized. The expectation is that as soon as SHA-3 is standardized then SHA-1 will be removed from the legacy use category. Therefore it is recommended that parties take immediate steps to stop using SHA-1 in legacy applications.

OBSERVATION: The literature also contains preimage attacks on a variant reduced to 45-48 rounds [18, 66].

3.3.3 Historical (non-endorsed) 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 [218, 313, 331–333]. Preimage resistance can also be broken in time $2^{124.4}$ [312].

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, [235]. In [211] further 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 guidelines for minimum parameter sizes for future and legacy use. For convenience these guidelines are summarised 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.

	Classification	
Primitive	Legacy	Future
HC-128	✓	✓
Salsa20/20	✓	✓
ChaCha	✓	✓
SNOW 2.0	✓	✓
SNOW 3G	✓	✓
SOSEMANUK	✓	✓
Grain	✓	✗
Mickey 2.0	✓	✗
Trivium	✓	✗
Rabbit	✓	✗
A5/1	✗	✗
A5/2	✗	✗
E0	✗	✗
RC4	✗	✗

Table 3.4: Stream Cipher Summary

3.4.1 Future Use Stream Ciphers

HC-128

HC-128 was an entrant to the eSTREAM competition and included in the final eSTREAM portfolio as promising for software implementations [353]. HC-128 uses a 128-bit key together with a 128-bit initialisation vector.

HC-256 uses 256-bit keys and initialisation vectors, is older dan HC-128, but was not submitted to the eSTREAM evaluation [352].

Salsa20/20 and ChaCha

Salsa20/r was an entrant to the eSTREAM competition [37]. It supports key lengths of 128 and 256 bits. The parameter r refers to the number of rounds used. Salsa20/12 was included in the final eSTREAM portfolio as promising for software implementations. The author of Salsa20 recommends to use the full 20 rounds for real applications.

The ChaCha stream cipher is a variant on the Salsa20 family. It modifies the Salsa design to obtain a better performance and increased diffusion. The ChaCha stream cipher forms the basis of the finalist to the SHA-3 hash function competition. The ChaCha cipher is used within the web browser Chrome.

OBSERVATION: Aumasson et al. report an attack on Salsa20/8 requiring 2^{251} encryptions and 2^{31} chosen IVs [21]. This also applies to the ChaCha family but decreased performance.

SNOW 2.0

SNOW 2.0 comes in a 128 and 256-bit key variants. The cipher is included in ISO/IEC 18033-4 [165]

OBSERVATION: A distinguishing attack against SNOW 2.0 is theoretically possible [266], but it requires 2^{174} bits of key-stream and work. A related-key attack exists on SNOW 2.0 with 256-bit key [195].

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.

SOSEMANUK

SOSEMANUK was an entrant to the eSTREAM competition and included in the final eSTREAM portfolio as promising for software implementations [34]. SOSEMANUK supports key lengths from 128 to 256 bits together with a 128-bit initialisation vector. The designers of SOSEMANUK don't claim more than 128 bits of security for any key length.

The literature contains several attacks on SOSEMANUK, which don't break the claim of 128-bit security. An attack requiring only a few words of key stream and with time complexity 2^{176} was shown in [122]. An attack requiring 2^{138} words of key stream and with time complexity 2^{138} was shown in [76, 213].

3.4.2 Legacy Stream Ciphers

Grain v1

Grain refers to a family of stream ciphers. The original version was an entrant to the eSTREAM competition [149]. After cryptanalysis [35], it was revised to Grain v1 [148]. The Grain v1 version supporting an 80-bit key and a 64-bit initialisation vector was included in the final eSTREAM portfolio as promising for hardware implementations. Grain 128, which is the version of Grain v1 with 128-bit key and 80-bit initialisation vector, is not endorsed. Instead, Grain 128a has been proposed recently [4]. At this time, it is too early to endorse the use of Grain 128a.

Mickey 2.0

Mickey 2.0 was evaluated by the eSTREAM competition and included in the final eSTREAM portfolio as promising for hardware implementations [22]. It uses an 80-bit key and an 80-bit initialisation vector. There exists also a scaled-up version Mickey-128 using 128-bit keys and initialisation values, but this version has not been officially evaluated by eSTREAM [22].

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 [165]. In [91] a distinguishing attack on Rabbit is described. The effect of this in practice has yet to be quantified thus for 2014 we downgrade Rabbit from suitable for future use, to only suitable for legacy use.

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 [168]. 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. [20] present a distinguishing attack with complexity 2^{30} on a variant of Trivium with the initialisation phase reduced to 790 rounds (out of 1152). Maximov and Biryukov [228] 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.

3.4.3 Historical (non-endorsed) Stream Ciphers

A5/1

A5/1 was originally designed for use in the GSM protocol. It is initialised 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 [26]. 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 [221]. 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 [222], and state recovery attacks [229]. (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 [344]. When initialised 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 [322].

Despite statistical biases being known since 1995, SSL/TLS does not discard any of the output bytes of RC4; this results in recent attacks by AlFardan et al. [6] and Isobe et al. [158].

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 guidelines for minimum parameter sizes for future and legacy use. For convenience these guidelines are summarised 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, $d \geq N^{1/2}$	$\ell(n) \geq 3072$ $e \geq 65537$, $d \geq N^{1/2}$
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 specialise 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 factorisation 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) [197]. 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 [184]). 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 [80].

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 [78, 79, 81]. 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. In keeping with the conservative nature of the suggestions in this report we suggest using $e = 65537$ for future systems using RSA signatures.

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 optimise 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 $N^{0.292}$ [53, 350]. Lattice attacks in this area have also looked at situations in which some of the secret key leaks in some way, see for example [115, 151]. We therefore advise 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. Indeed, if standard practice is followed and e is selected first then d will be of approximately the same size as N with overwhelming probability.

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 parametrised 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. $p = 2, 3$ a new algorithm was presented early 2013 by Joux [181], 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 [24]. Also in 2013 a series of record breaking calculations were performed by a French team and an Irish team for characteristic two fields, resulting in the records of $\mathbb{F}_{2^{6120}}$ [135] and $\mathbb{F}_{2^{6168}}$ [179]. For characteristic three the record is $\mathbb{F}_{3^{582}}$ [338]. For prime values of n the best result is a discrete logarithm calculation in the field $\mathbb{F}_{2^{809}}$ [57]. All of these results make use of special modification to the function field sieve algorithm [3]. In light of these results no system should be deployed relying on the hardness of the DLP in small characteristic fields. It is for this reason that we impose the condition $\ell(p) > 256$ (resp. $\ell(p) > 160$ for legacy systems) in Table 3.5.

For large prime fields, i.e. $n = 1$, the algorithm of choice is a variant of the Number Field Sieve [132]. The record here is for a finite field $\mathbb{F}_p$ with p a 530 bit prime [196] set in 2007. In light of the "equivalence" between the number field sieve for factoring and that for discrete logarithms our advise 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 [180, 182]. 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 according to our advise below have $\log_2(p) \geq 160$.

ECDLP

Standard elliptic curve cryptography (i.e. ECC not using pairings) comes in two flavours in practice, 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 [237].
- 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 [315, 321, 329].
- If $p = 2$ then n should be prime. This is to avoid so-called Weil descent attacks [129].

The above three conditions are denoted by $\star$ in Table 3.5. 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 $p = 2$. To avoid implementation mistakes in protocols we *strongly* advise that curves are selected with $h = 1$. Some fast implementations can be obtained when $h = 4$, but when using these protection against small subgroup attacks need to be also implemented.

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. [116] 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 no weakness is known for such curves. If curves are selected at random it is over whelmingly likely that the curve has a large endomorphism ring in any case.

The largest ECDLP records have been set for the case of $n = 1$ with a p of size 109-bits [56], and for $p = 2$ with $n = 109$ [70]. These record setting achievements are all performed with the method of distinguished points [340], which is itself based on Pollard's rho method [282]. 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. [13, 14, 320] specify a set of recommended curves; many of which also occur in other standards and specifications, e.g. in TLS [48]. 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 utilise 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. In addition, note that the conditions in Table 3.5 for pairings immediately imply all the special conditions for elliptic curve based systems indicated by a $\star$ in the ECDLP row. This explains the lack of a $\star$ in the pairing row of Table 3.5.

3.6 Key Size Analysis

Providing 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 providing key sizes for specific application domains we make an *implicit* assumption that the primitive, scheme or protocol which utilises this key size is not broken in the near future. All primitives 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 providing key size guidelines 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 guidelines 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 guidelines. The reader will see that we have essentially followed the NIST equivalence [262] 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 guidelines; clearly this offers a good long term security guarantee. It is plausible that similar advise 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 *guidelines for future use* can be summarised in the following simple choices:

1. Block Ciphers: For near term use we advise AES-128 and for long term use AES-256.

	Parameter	Legacy	Future System Use	
			Near Term	Long Term
Symmetric Key Size	k	80	128	256
Hash Function Output Size	m	160	256	512
MAC Output Size	m	80	128	256*
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 Analysis. A * notes the value could be smaller due to specific protocol or system reasons, the value given is for general purposes.

2. Hash Functions: For near term use we advise SHA-256 and for long term use SHA-512.

3. Public Key Primitive: For near term use we advise 256 bit elliptic curves, and for long term use 512 bit elliptic curves.

Note, that all of our guidelines need to be read given the aspects described in Section 2.4 which we do not cover in this report. Finally, we note that the guidelines above, and indeed all analysis in this document, is on the basis that there is no breakthrough in the construction of quantum computers. If the development of quantum computers became imminent, then all this documents guidelines would need to be seriously reassessed. In particular *all* of the public key based primitives in this document should be considered to be insecure.

Chapter 4

Basic Cryptographic Schemes

As mentioned previously 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 idealised objects. A similar issue occurs with some proofs using idealised groups (the so-called *generic group model*), or idealised 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 [130]. As another example one can imagine using an RSA private key as both a decryption key and as a key to generate RSA signatures; indeed this latter use-case is permitted in the EMV chip-and-pin system [93]. 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 [9,93,271], and simply warn the reader to violate the key separation principle with extreme caution. In general key separation is a good design principle in systems, which can help to avoid logical errors in other system components. If key separation is violated then we advise this is only done following a rigorous analysis, and associated security proofs.

In Tables 4.1, 4.2, 4.4 and 4.5 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 guidelines 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* advise 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. A scheme for which it is *probably true* that it is IND-CVA is marked with a bracketed tick. 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 [299].

Many modes make use of either a nonce or an random IV. A *nonce* is a *number used once*, it is a non-repeating value but not necessarily random. Thus a nonce could be a non-repeating sequence number. On the other hand a random IV should be random, and unpredictable to the adversary.

4.1.1 ECB

Electronic Code Book (ECB) mode [254] 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

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	✓	✓	✓	
EAX	✓	✓	✓	An improved version of CCM
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

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 [254] 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. With a non-random or predictable IV, CBC mode is insecure. In particular using a nonce as the I is insufficient to prove security.

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 [272, 342, 354]. Again usage of CBC within an authenticated encryption scheme (and utilising uniform error reporting in the case of Encrypt-then-MAC schemes) can mitigate against such attacks.

4.1.3 OFB

Output Feedback (OFB) mode [254] 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 [254] 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 [8], if the underlying block cipher is secure and the IV is random (i.e. not a nonce).

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 [254] 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.

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.

4.1.6 XTS

XTS mode [257] is short for *XEX Tweakable Block Cipher with Ciphertext Stealing* and is based on the XEX tweakable block cipher [297] (using two keys instead of one). The mode was specifically designed for encrypted data storage using fixed-length data units, and was 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 [257] 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 [299, Chapter 6] and [220]. The underlying tweakable block cipher XEX is proved secure as a strong pseudo-random permutation [297].

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 pseudo-random 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), hash function based schemes (Section 4.2.2), and those based on universal hash functions (Section 4.2.3). Before looking at specific constructions we note that a MAC function with security 2^s should have an output size of at least s bits; and for a well designed MAC function the output size should be exactly s bits. If we truncate a MAC output by ϵ percent, then the security drops to $2^{\epsilon \cdot s}$ for a well designed MAC function.

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

Table 4.2: Symmetric Key Based Authentication Summary Table. When instantiating the primitives they should be selected according to our division into legacy and future use to provide the MAC function with the same level of security.

h!tb

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

Table 4.3:

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 [170] 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 utilised with any cipher; one of which uses a non-standard processing of the first block. Table 4.3 summarises these six algorithms, 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 $1, \ldots, q-1$. In schemes that use extra keys K', K'', all keys are derived from a single key in a way specified by the standard. Usually there is no corresponding increase in security if these keys are generated independently.

We treat here EMAC, AMAC and CMAC, being the most utilised variants. Note that vanilla CBC-MAC is on its own not considered secure, except in very limited circumstances; for example where the message length is pre-pended to the message before applying the MAC function. Of course all MACs should be used with keysizes and output sizes which match our key size proposals. In particular this means for high security levels, i.e. equivalent to 256-bits of AES security, that these MAC functions cannot be used with AES as their output lengths are limited to the block length of the underlying block cipher, unless the application allows shorter MAC values for some reason. Shortened MAC outputs may be secure if the length that the MAC key is relatively short lived. Thus the MAC can only be verified for a short length of time. Compare this with encryption, where we want to ensure security for a long time, even if the key is short lived.

EMAC

The Algorithm was introduced in [274] and is specified as Algorithm 2 in ISO-9797-1 [170]. 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. For a variant of the scheme that uses two independent keys, provable security guarantees have been derived in [274,276]. Note however that the security of the scheme is bounded by 2^k, where k is the length of a single key. 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 [11] and is also specified as Algorithm 3 in ISO 9797-1 [170]. 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 [172] and standardized as Algorithm 5 in [170]. It enjoys provable security guarantees under the assumption that the underlying block-cipher is a PRP [242]. 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. (This is a problem if Key Check Values as defined in ANSI X9.24-1:2009 [12] are used.)

4.2.2 Hash Function Based MACs

HMAC

The HMAC scheme[1] was introduced in [65] and standardized in [171, 203, 250]. The construction is based on an underlying hash function which, itself, needs to have an iterative design of the Merkle–Damgård form [90, 238]. Provable security results for HMAC aim to establish that HMAC is a PRF [28, 65]. Interestingly, this can be done only relying on the pseudo-randomness of the underlying compression-function and does not require collision-resistance [28]. In particular, this means that instantiations of HMAC with compression-functions that are not collision-resistant may still

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

be reasonably secure, provided that the collision attacks do not yield distinguishing attacks against the pseudo-randomness of the underlying compression-function. HMAC-MD4 should therefore not be used while HMAC-SHA1, HMAC-MD5 are still choices for which forgeries cannot be made. However, we do not propose usage with MD-5 even for legacy applications and use with SHA-1 is proposed with the usual caveats mentioned before. Conservative instantiations should consider HMAC-SHA2 and HMAC-SHA3.

4.2.3 MACs Based on Universal Hash functions

A universal hash function is actually a family of hash functions [68]. The properties of a universal hash function are defined over the distribution of all hash functions in the family. This means that it becomes possible to define a property like *collision probability* in a mathematically meaningful way: The probability that two inputs give a collision is defined fraction of functions in the family for which two inputs result in the same output.

Universal hash functions can be used in MAC constructions with provable security properties. A hash function from the family is fixed, and then on each invocation of the hash function a one-time (or pseudo-random) pad is added to the output. This effectively means that on each invocation, a new hash function is defined in a way that is unpredictable by the attacker. In cryptographic applications, this is typically achieved by a combination of a secret key (defining the element of the family) and a non-repeating value or nonce (defining the pad). For some constructions, re-use of the same nonce leads to recovery of the secret key.

UMAC

UMAC was introduced in [47] and specified in [208]. The scheme has provable security guarantees [47]. 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 Handschuh and Preneel [143], the 32-bit output version results 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. In any case MAC tags of 64-bits in length should be used in all cases.

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 altered 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, 287, 307]. Due to these potential

issues we leave the use of GMAC outside of GCM mode in the legacy only division. See the entry on GCM mode below for further commentary.

4.3 Authenticated Encryption (with Associated Data)

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 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).

Authentication Encryption with Associated Data (AEAD) [296] is an extension of AE. In an AEAD scheme there exists extra associated data, such as a header, which is authenticated but not encrypted. As a result AEAD more closely captures how AE is used in practice. All of the modes we describe in this section are AEAD schemes, with the exception of generic composition which depends on the exact construction.

4.3.1 Generic Composition (Encrypt-then-MAC)

Encrypt-then-MAC is probably the simplest mechanism to construct an authenticated encryption scheme. The security of the method was studied in [31], 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 to be used, due to the basic requirement of probabilistic encryption in most applications.

Despite the well known analysis of [31] of Encrypt-then-MAC being the best choice, sometimes the result can be misinterpreted. The result of [31] says the Encrypt-then-MAC method is secure if the encryption scheme is a *probabilistic* IND-CPA scheme and the MAC function is UF-CMA secure. The ISO 19772 standard builds an Encrypt-then-MAC scheme from a nonce-based encryption scheme (one which is not IND-CPA) and then appeals to [31] to claim security. The key difference is that (say when using CBC or CTR mode) the IV is not authenticated, this was pointed out in [241], and changes are being made to the ISO standard to correct this bug.

Other related constructions, 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 [7]. 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), see [204]. However, the probability of an error being made in the choice, implementation or application are too large to enable safe usage. Further details on how IV

and nonce-based constructions of this type may be composed securely can be found in the paper by Namprempre et al. [241].

4.3.2 OCB

Offset Codebook (OCB) mode [167] was proposed by Rogaway et al. [301]. 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 [300].

4.3.3 CCM

CCM mode [255] was proposed in [349] 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 [176], and a critique has been given in [303].

The main drawback of CCM mode comes from its inefficiency. 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 [167] was presented in [33], 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 [202]. 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. Care should be taken to ensure that IVs are never repeated otherwise forgery attacks may be possible. 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) [256] was designed by McGrew and Viega [230, 231] 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 [173] 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 [178] demonstrated a problem in the NIST specification of GCM when non-default length IVs are used. Ferguson's [177] critique highlights a security weakness when short authentication tags are used. To prevent attacks based on short tags it is wise to insist that authentication tags have length at least 96 bits. Furthermore it is wise to also insist that the length of nonces is fixed at 96 bits. Saarinen [307] raises the issues of weak keys which may lead to cycling attacks. The work of Proctor and Cid [287] presents an algebraic analysis which demonstrates even more weak keys. In the conclusion of their paper Proctor and Cid discuss the significance of weak key attacks. They state that although it is highly undesirable for almost every subset of the keyspace to be a weak key class, for many schemes (GCM included) this will not reduce the security to an unacceptable level.

4.4 Key Derivation Functions

Key Derivation Functions (KDFs) are used to derive cryptographic keys from from a source of keying material, such as a shared random strings (in the case of key agreement protocols) or from an entropy source (in the case of key generation). 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. Often they take additional input of a shared info field, which is not necessarily secret.

The idea is that the input keying material to the KDF may reveal some partial information, may not be uniformly generated, may have some statistical bias, etc. The KDF takes the input and outputs a pseudo-random key from the imperfect input source of semi-secret randomness. An additional usage is to expand a given cryptographically strong key into multiple keys. Thus the KDF as as both a randomness extractor as well as an expander. See [205] for a extensive discussion on the extract-then-expand approach to KDF design; and HKDF in particular.

In security proofs KDFs are often modelled as random oracles; but simply instantiating them with a vanilla hash function is not to be used (despite this being common practice in academic papers). In practice KDFs are specifically designed, each of which is built upon a specific primitive such as a keyed PRF (i.e. a MAC function) or a hash function. When instantiating a KDF the underlying primitive (PRF or hash function) is assumed to be secure. In other words one should select one of our selected MAC or hash functions as appropriate. We summarize the constructions in Table 4.4, where the column "Building Block" refers to the underlying primitive used to create

the KDF primitive.

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

Table 4.4: Key Derivation Function Summary Table. When instantiating the primitives they should be selected according to our division into legacy and future use to provide the PRF function with the same level of security.

4.4.1 NIST-800-108-KDF

NIST-SP800-108 [251] defines a family of KDFs based on pseudo-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 [261].

4.4.2 X9.63-KDF

This KDF is defined in the ANSI standard X9.63 [14] 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, however there are now better designs for KDF's than this one. We still include it for future use however, as there are no reasons (bar the existence of better schemes) to degrade it to legacy only.

4.4.3 NIST-800-56-KDFs

A variant of the X9.63-KDF is defined in NIST-SP800-56A/B, [259, 260]. 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). Similar comments apply to its use for future and legacy systems as that made for X9.63-KDF above.

In NIST-SP800-56C [261] 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 [251], 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 HKDF, IKE-v1-KDF and IKE-v2-KDF

HKDF, presented in [205] and [206] is a KDF based on the HMAC function. If forms the basis of the design of the KDFs specified in [145] and [191] for the IKE sub-protocol of IPsec. In all 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. We can find no public explanation of this decision but we expect this is due to the analysis in [75].

4.4.5 TLS-KDF

This is the KDF defined for use in TLS, it is defined in [94] and [48]. 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 aide mémoir 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 proposed 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. Most public key encryption schemes either deployed or in standards follow the KEM/DEM hybrid encryption paradigm. 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 [278,279] has no modern security proof, although it is used in the SSL/TLS protocol extensively. A chosen ciphertext reaction attack[2] [49] 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 [81,84,146]. Attacks on various cryptographic devices which use this encryption scheme have also been reported [25]. This method of encryption should not be used for any applications, bar the specific use (for legacy reasons) in SSL/TLS. The specific use within modern versions of SSL/TLS has been shown to be provably secure [207], however this usage is *not* forward secure so even usage in SSL/TLS should be phased out as soon as possible. The current proposal is for this scheme to be removed in the forthcoming TLS 1.3 standard.

[2] A type of chosen ciphertext attack in which the attacker obtains valid/in-valid ciphertext signals as opposed to full decryptions for his chosen ciphertexts

Scheme	Classification Legacy	Classification 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	✗	✗	
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
(EC)KDSA	✓	✓	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)RDSA	✓	✗	Weak provable security guarantees
ISO-9796-2 RSA-DS1	✗	✗	Attack exists (see notes)

Table 4.5: Public Key Based Scheme Summary Table

4.6.2 RSA-OAEP

Defined in [279], and first presented in [32], 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 [126], and the proof has been verified in the Coq theorem proving system [27]. A decryption failure oracle attack is possible [223] 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 good practice to ensure 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 [164], 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 (modelling the KDF as a random oracle), with very good security guarantees [147,324]. We assume that the KDF used in the scheme be one of the good ones mentioned in Section 4.4.

4.7.2 PSEC-KEM

This scheme is defined in [164] and is based on elliptic curves. Again when modelling 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 we assume that the KDF used in the scheme be one of the good ones from Section 4.4.

4.7.3 ECIES-KEM

This is the discrete logarithm based encryption scheme of choice. Defined in [14, 164, 319], 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 we assume that the KDF used in the scheme be one of the good ones from Section 4.4.

4.8 Public Key Signatures

4.8.1 RSA-PKCS# 1 v1.5

Defined in [278, 279] 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 propose this be used beyond legacy systems.

4.8.2 RSA-PSS

This scheme, defined in [279], can be shown to be UF-CMA secure in the random oracle model [175]. 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 [82, 83, 185], but is not wise to 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 utilised 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 [169] 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 [85, 86] which reduced breaking the padding scheme from 2^{80} operations to 2^{61} operations. Using a number of implementation tricks the authors of [86] manage to produce forgeries in a matter of days utilising 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 randomisation 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 propose 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 [13, 249, 319]; and there exists a number of variants including the German DSA (GDSA) [152, 161], the Korean DSA (KDSA) [161, 337] and the Russian DSA (RDSA) [133, 162]. 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 (bar KDSA) have weak provable security guarantees; whilst some proofs do exist they are in less well understood models (such as the generic group), for example [61]. 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.

The KDSA algorithm uses a hash function to compute the r-component of the signature, a full proof in the random oracle model can be given for this variant [59]. Thus KDSA falls into our category of suitable for future use. KDSA also has a simpler signing equation than DSA, it does not require a modular inversion, however the extra hash function invocation is likely to counterbalance this benefit.

All (EC)DSA variants also suffer from lattice attacks against poor ephemeral secret generation [154, 245, 246]. A method to prevent this, proposed in [293] 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. This however needs to be used with extreme caution as the use of a deterministic emphemeral key derivation technique could lead an implementation open to side-channel analysis.

4.8.6 PV Signatures

ISO 14888-3 [161] 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 [281], and the scheme is often denoted as the PV signature scheme[3]. The PV signature scheme 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.

[3]There is another PV signature scheme which this should not be confused with due to Pintsov and Vanstone [277] which is a signature scheme with message recovery originally used to secure electronic postal franks.

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

4.8.7 (EC)Schnorr

Schnorr signatures [318], standardized in [162], are like (EC)DSA signatures with two key differences; firstly the signing equation is simpler (allowing for some optimisations) 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 [280]. There is also a proof in the generic group model [244]. In addition the signature size can be made shorter than that of DSA. 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 defences proposed for (EC)DSA signatures should also be applied to Schnorr signatures.

Chapter 5

Advanced Cryptographic Schemes

In this chapter we discuss more esoteric or specialised schemes. These include password based key derivation, password based encryption, key-wrap algorithms and identity based encryption. We summarize our conclusions in Table 5.1.

Scheme	Categorisation		Notes
	Legacy	Future	
Password Based Key Derivation			
PBKDF2	✓	?	See text
bcrypt	✓	?	See text
scrypt	✓	?	See text
Key Wrap Algorithms			
KW	✓	✗	No security proof; no associated data
TKW	✓	✗	No security proof; no associated data
KWP	✓	✗	No security proof; no associated data
AESKW	✓	✗	No security proof; inefficient
TDKW	✓	✗	No security proof; inefficient
AKW1	✓	✗	No security proof; no associated data
AKW2	✗	✗	Not fully secure
SIV	✓	✓	See text
Identity Based Encryption			
BB	✓	✓	See text
SK	✓	✓	See text
BF	✓	✗	See text

Table 5.1: Advanced Scheme Summary Table

5.1 Password-Based Key Derivation

Section 4.4 provides details on algorithms to derive cryptographic keys from a secret random string. In many situations the only secret that may be present is a password but due to their low entropy and possibly poor randomness they need to be used with special care and must not be used directly as cryptographic keys. As a result a special key-derivation function should be used which is designed for this case. Password-Based Key Derivation functions are a very important topic since passwords are still the main mechanism by which humans interact with cryptographic services. There exists some standardisation of these functions by NIST [253], and ISO is currently writing the ISO/IEC 11770-6 standard [160].

For all the algorithms we detail below there exists no formal security analysis and so we only give classifications for legacy use at this time. While there exists no known vulnerabilities in any of the schemes we do not make any statements as to their future use until more formal provable security results are known. Given a password derived key a password based encryption algorithm can be obtained by applying a block cipher with the associated key, see [186] for an example of this.

5.1.1 PBKDF2

NIST SP 800-132 [253] standardises the PBKDF2 function, which was first defined in RFC 2898 [186]. PBKDF is based on any secure PRF; in [186] it is defined with HMAC using SHA-1. Additionally, PBKDF2 is defined by an iteration count which specifies the number of times the PRF is iterated. The iteration count is used to increase the workload of dictionary attacks and should be as large as possible whilst ensuring the compute time is not unnecessarily long. A minimum of 1000 iterations is proposed.

The input to the key-derivation function is the password, a salt and the desired key length. The salt is used to generate a large set of keys for each password. It should be generated with a secure random number generator (cf. Section 6.2) and be at least 128 bits long. The key length should be at least 112 bits.

Despite the ability to adjust the number of iterations it is still possibly to implement dictionary attacks relatively cheaply on ASICs or GPUs. The bcrypt function and scrypt functions provide progressively greater resistance to such attacks due to the respective attacks increasing need for additional RAM.

5.1.2 bcrypt

bcrypt was designed by Provos and Mazières [289]. It is based on the blockcipher Blowfish (cf. Section 3.2.2). bcrypt is more resistant to dictionary attacks than PBKDF2.

5.1.3 scrypt

scrypt [273] was designed by Colin Percival to create a key derivation function which was much more resistant to dictionary attacks than bcrypt. The scheme was introduced in 2009 and so is much younger than other schemes meaning it has not been subject to as much usage and analysis.

5.2 Key Wrap Algorithms

In this section we discuss the main modes of operation for using a block cipher to wrap other keys. This functionality is particularly important for the storage and transmission of symmetric keys. An important consideration when using key wrap, is that the security level of the key wrap is bound by the key length of the key that is used to encrypt. For instance, wrapping an AES-256 key under an AES 128-bit key will reduce the security of the AES-256 key to 128 bits (or less).

The accepted security notion for key wrap is deterministic authenticated encryption. It is related to authenticated encryption (Section 4.3), in particular there is an important (practically relevant) notion of binding associated data with the encrypted key (for example key usage information). The majority of modes for key wrap lack formal analysis. For a detailed discussion of the key wrap security notion and a critique of several key wrap modes, refer to [302].

5.2.1 KW and TKW

The two schemes AES Key Wrap, abbreviated KW, and Triple DEA Key Wrap, abbreviated TKW, are specified in NIST Special Publication 800-38F [258]. RFC 3394 [316] and ISO/IEC 19772 [166] both contain an equivalent specification of AES Key Wrap. The schemes KW and TKW do not natively support associated data.

Both KW and TKW are constructed using two transformations. The first transformation creates a variable input length cipher from the block cipher. The input lengths of the cipher is measured in semi-blocks (with a minimum of three semi-blocks). Thus for key wrap based on AES strings with bit length a multiple of 64 can be encrypted, whereas for 3DES the input length needs to be a multiples of 32 bit. To encrypt n semi-blocks, $6(n-1)$ blockcipher calls are needed, which is a relatively high overhead. There are no formal results regarding the security of the variable input length cipher.

The second stage is the use of the variable input length cipher to create a deterministic authenticated encryption scheme. For both schemes this is achieved by padding the message with a fixed integrity check value, that is checked upon decryption. This method is provably secure [302].

Since 3DES should be considered legacy only, so should TKW. KW can be still be used in scenarios where there is no associated data.

5.2.2 KWP

The scheme AES Key Wrap with Padding, abbreviated KWP, is specified in [258] and RFC 5649 [153]. It shares the variable input length cipher from KW, but due to the use of an explicit padding scheme, inputs of any number of octets are allowed.

5.2.3 AESKW and TDKW

The two key wrap schemes AESKW and TDKW are specified in [10]. They share the variable input length ciphers from KW and TKW, respectively. The padding scheme allows key data of arbitrary bit length. Additionally, the padding scheme natively supports associated data to be authenticated, but it should be noted that for authentication, this data is encrypted it along with the actual payload.

5.2.4 AKW1

The scheme AKW1 is specified in ANSI X9.102 [10] and consists essentially of a SHA1 based padding scheme, followed by two layers of CBC encryption, one with a random IV and one with a fixed IV, where the underlying blockcipher is 3DES. The random IV makes the scheme probabilistic, making classification as an authenticated encryption scheme (without associated data) more accurate than as a key wrap scheme. Even when instantiated with a modern block cipher instead of 3DES, AKW1 should be considered a legacy only construction.

5.2.5 AKW2

The scheme AKW2 is specified in ANSI X9.102 [10] and corresponds to an Encrypt-then-MAC scheme using related keys. For the encryption, CBC mode using TDEA is stipulated, whereas for authentication CBC-MAC is used. The scheme supports associated data and indeed, the first block of associated data is used as initialisation vector for the CBC mode. AKW2 is demonstrably not a secure key wrap scheme [302] and we believe it should not be used.

5.2.6 SIV

Synthetic Initialisation Vector (SIV) authenticated encryption was introduced by Rogaway and Shrimpton [302]. It is a 2-pass mode based on using an IV-based encryption scheme with a pseudo-random function. The pseudo-random function is used to compute a tag that is used both for authentication purposes and as IV to the encryption scheme. SIV is captured by RFC 5297 [144], combining CMAC with AES in counter mode. SIV is provably secure and relatively efficient.

5.3 Encrypted Storage

We will also discuss recent innovations in related to cloud data storage in this field; in particular related to de-duplication of encrypted data.

5.4 Identity Based Encryption/KEMs

5.4.1 BF

An identity based encryption (IBE) scheme allows a user to encrypt to a public consisting of an arbitrary string. This string can be an identity, identifier or more generally any string meaningful to the user. To enable decryption a trusted authority issues decryption keys associated to the strings to users. As such identity based encryption provides a key escrow service by default. The "gold" standard for security is that a scheme should be indistinguishable against an adversary who can request secret keys for arbitrary identities (bar the target one), and can also request decryptions of arbitrary ciphertexts with respect to any identity (bar the target identity). This is the ID-IND-CCA security model.

A lot of advanced encryption functionalities can be built from these ideas; e.g. hierarchical IBE, functional encryption. Many of the more academic schemes are based on the idea of a Water's Hash, which first appeared in [348]. In this paper an IBE scheme which is secure in the standard model is given. No standardized scheme however uses this latter construction.

The Boneh–Franklin IBE scheme [54, 55] is known to be ID-IND-CCA secure in the random oracle model and is presented in the IEEE 1363.3 standard [155]. The scheme is not as efficient as the following two schemes, and it does not scale well with increased security parameters; thus it we only categorise it for legacy use. The underlying construction can also be used in a KEM mode.

5.4.2 BB

The Boneh–Boyen IBE scheme [52] 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 [155], 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.

5.4.3 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 [72] and a KEM construction [73], and is also defined in the IEEE 1363.3 standard [155]. 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 [74], on q-style assumptions.

Chapter 6

General Comments

In this chapter we discuss a number of general issues related to the deployment of cryptographic primitives and schemes. In this edition of the report we restrict ourselves to hardware and software side-channels, random number generation and key life-cycle management.

6.1 Side-channels

Traditionally, cryptographic algorithms are designed and analysed in the black-box model. In this model, an algorithm is merely regarded as a mathematical function that will be applied to some input to generate some output, regardless of implementation details. An evaluation of a keyed algorithm in the black-box model assumes that an adversary knows the specification of the algorithm and can observe pairs of inputs I and outputs $O = E_k(I)$ of a black box implementing the algorithm.

When cryptography is implemented on embedded devices, black-box analysis is not sufficient to get a good picture of the security provided. The cryptographic algorithms are executed on a device that is in the possession and under the physical control of the user, who may have an interest in breaking the cryptography, e.g. in banking applications or digital right management applications. The physical accessibility of embedded devices allows for a much wider range of attacks against the cryptographic system, not targeting the strength of the algorithm as an abstract mathematical object, but the strength of its concrete implementation in practice.

Classical examples for side-channels include the execution time of an implementation [200], the power consumption of a chip [201] and its eletromagnetic radiation [128]. More exotic examples include acoustics [19], temperature [60] and light emission [328]. Some side-channels can be observed only by means of an invasive attack, where the computing device is opened. Others can be observed in a passive attack, where the device is not damaged.

There are many reasons for side-channel leakage, including hardware circuit architectures, micro-architectural features and implementations. Interestingly, many side-channels arise from optimisa-

tions. For example, circuits in modern CMOS technology consume power only when the internal state changes. The amount of power consumed is proportional to the number of state bits that change. This is clearly a side-channel. For other examples of the relation between optimisation and side-channels, please see Section 6.1.1.

Many cryptographic algorithms are constructed as *product ciphers* [323]: one or a few cryptographically weak functions are iterated many times such that the composition is secure. Other algorithms use a small number of complex operations. In order to implement such algorithms, however, these complex operations are usually broken down into sequences of less complex operations. Hence, their implementations are similar to product ciphers. Furthermore, in keyed algorithms (or their implementations), typically the key is introduced gradually: the dependence of the intermediate data on the key increases in the course of the algorithm (or implementation).

Side-channel attacks capitalise on this property of gradually increasing security. While it is (supposedly) hard to attack the full cryptographic algorithm, it is much easier to attack the cryptographically weak intermediate variables. Depending on the side-channel, measurements of the leakage contain information about the intermediate variables at each instance of time (e.g. power consumption), or about an aggregate form thereof (e.g. execution time). Thus, side-channel measurements allow to zoom in on the algorithm and to work on a few iterations only of the cryptographically weak functions. By working with intermediate variables that depend only on a fraction of the bits of the secret key, side-channel attacks allow to apply a divide-and-conquer strategy.

6.1.1 Countermeasures

Countermeasures against side-channel attacks can be classified into two categories. In the first category, one tries to eliminate or to minimise the leakage of information. This is achieved by reducing the signal-to-noise ratio of the side-channel signals. In the second category, one tries to ensure that the information that leaks through side-channels, cannot be exploited to recover secrets. Typically, one will implement a combination of countermeasures. Increasing the key size will (in general) not improve the resilience against side-channel attacks.

Constant-time algorithms

The first academic publication of a physical attack is the timing attack on RSA [200]. In a naive implementation of modular exponentiation, the execution time depends on the value of the exponent, i.e. the private key. By observing the execution time of a series of decryptions or signatures, an adversary can easily deduce the value of the private key.

Hence, a first countermeasure to be taken is to ensure that the execution time of the cryptographic algorithm doesn't depend on the value of secret information. The difficulty of this task depends greatly on the features of the processor that the software will run on and the compiler that is being used to translate high-level code into low-level assembly instructions.

Simple processors and low-level programming languages give the programmer absolute access

to the control flow of the program, making it possible to write code that executes in constant time. Modern pipelined processors contain units for branch prediction, out-of-order execution and other systems that may complicate the task of predicting the exact execution time of an algorithm or a subroutine. These units may interact with compiler options and settings in ways that are difficult to fully understand. In such environments, it may be difficult to achieve 100% constant-time code.

Observe that constant-time code is usually slow code. Indeed, any optimisation that can be applied only for a fraction of the values that a secret variable can take, leads to non-constant execution time and therefore has to be excluded.

Constant power consumption

For implementations in hardware, constant execution time is usually easy to achieve. However, the side-channels of hardware implementations typically leak much more information the side-channels of software implementations. For example, the instantaneous power consumption signal not only leaks the execution time of the algorithm, but also its level of activity at each instant of time. Balanced circuits reduce the signal leaking from hardware implementations [334].

Reduce secret data dependent branches

Branching instructions where the condition depends on the value of secret data are an obvious cause for differences in execution time. Since it is difficult to ensure that all branches execute in exactly the same time, it is recommended to prefer methods that have fewer data dependent branches. For example, instead of implementing an exponentiation by means of square-and-multiply (or double-and-add) techniques, one can employ the Montgomery ladder method, which behaves very regularly [183].

Reduce secret data dependent lookups

Modern processors can execute instructions much faster than modern main memories can deliver new instructions and operands. In order to avoid that processors have to wait, memories are organised in a hierarchy. At the bottom are the very large and very slow disks. Above are layers of increasingly smaller and faster memory units: RAM, L2 cache, L1 cache. This memory architecture has as side-effect that the time it takes to lookup data, is not constant. If the data is present in L1 cache, then the lookup goes faster than if it needs to be brought in from L2 cache or RAM.

Many implementations of cryptographic algorithms use lookup tables. Unless special precautions are taken, these lookup tables will not be present in L1 cache at the start of the execution of the algorithm. Sometimes the tables don't even fit into the L1 cache. This usually causes differences in execution time, which may lead to timing attacks [36, 336].

Bit-slice implementations are implementations that avoid table lookups. Instead they compute table elements on the fly [39]. In particular if the algorithm applies the same function to several parts of the input in parallel (SIMD parallelism), the performance of bit-slice implementations may

be very competitive to table-based implementations [190]. For the specific case of AES (and other algorithms using the AES S-box), the AES-NI instructions can be used to avoid table lookups.

Masking

The purpose of masking is to ensure that the value of individual data elements is uncorrelated to secrets. Hence, if there is leakage on the value of individual data elements, this will not lead to recovery of the secrets. Clearly, if an attacker can combine signals of different elements, he can again start to recover the secrets, but the approach can be generalised to higher levels, making tuples, triplets, ... of data independent of the value of the secret [288]. Masking can be done for software implementations and for hardware implementations.

In hardware, masking can be employed at gate level [157, 335], at algorithm level [5], or in combination with circuit design approaches [283].

The Threshold Implementation method is a masking approach that achieves provable security based on secret sharing techniques at a moderate cost in hardware complexity [247, 284]. It can also be used to mask software implementations. An alternative approach based on Shamir's secret sharing scheme is presented in [134].

6.2 Random Number Generation

Randomness is needed in almost all cryptographic systems and protocols. For example, random numbers are needed for generating asymmetric key-pairs, for defining symmetric keys, for generating initialisation vectors (IVs) for cryptographic modes of operation, in challenge-response protocols, as additional inputs to most standardised public key encryption and signature algorithms, and to generate ephemeral values in key exchange protocols. The existence of suitable random sources is taken for granted in much of the research literature in cryptography, and almost all formal security analysis of cryptographic schemes fails if perfect randomness assumptions are not met. Yet there are many prominent examples of randomness failures with severe security consequences; examples include:

- Netscape's implementation of SSL, which was discovered in 1996 to make use of a random number generator in which the only sources of entropy used to seed the generator were the time of day, the process ID and the parent process ID [131].
- The Debian OpenSSL randomness failure, in which a patch applied by a Debian developer led to substantially reduced entropy being available for key generation in OpenSSL [92]. Affected keys included SSH keys, OpenVPN keys, DNSSEC keys, and key material for use in X.509 certificates and session keys used in SSL/TLS connections, with all keys produced between September 2006 and May 2008 being potentially suspect.

- Two independent analyses of public keys found on the Internet [150, 215], which discovered, amongst other things, that many pairs of RSA public keys had common factors, making the derivation of the corresponding private keys a relatively trivial matter. The identified issues are at least in part attributable to poor randomness generation procedures, especially in the Linux kernel [150]. A follow-up study on a particular smart-card deployment involving RSA is reported in [38].
- Ristenpart and Yilek studied how randomness is handled across virtual machine resets [293], discovering that the state of the PRNG can often be predicted to the point where an attack against a DSA signing key can be mounted in the context of TLS (two signatures on distinct messages being produced with the same random input leading to immediate recovery of the DSA private key).

6.2.1 Terminology

We refer to Random Number Generators (RNGs), but these are also often referred to as Random Bit Generators in the literature. A suitable source of random bits can always be turned into a source of random numbers that are approximately uniformly distributed in a desired range by various means (see [123, Section 10.8], [264, Appendix B] for extensive discussion of this important practical issue). In what follows we make extensive reference to the NIST standard [264], however the interested reader should also consult the ISO 18031 standard [163] and ANSI X9.82 [15].

We distinguish between True Random Number Generators (TRNGs) and Pseudo-Random Number Generators (PRNGs). TRNGs usually involve the use of special-purpose hardware (e.g. electronic circuits, quantum devices) followed by suitable post-processing of the raw output data to generate random numbers. In an ideal world, all random number requirements would be met by using TRNGs. But, typically, TRNGs operate at low output rates (relative to PRNGs) and are of moderate-to-high cost (relative to PRNGs which are usually implemented in software). A TRNG device might be used to generate highly sensitive cryptographic keys, for example system master keys, in a secured environment, but would be considered "overkill" for general-purpose use. PRNGs are suitable for general-purpose computing environments and usually involve a software-only approach. Here, the approach is to deterministically generate random-looking outputs from an initial seed value. We note that NIST [264] refer to PRNGs as DRBGs, where "D" stands for "deterministic", stressing the non-random nature of the generation process. Here, we focus on PRNGs, since TRNGs do not in general offer the flexibility and cost profile offered by (software) PRNGs.

A PRNG usually includes a capability for reseeding (or refreshing) the generator with a fresh source of randomness. The problem of obtaining suitable and assured high-quality randomness for the purposes of reseeding is one of the most challenging aspects of designing systems that use PRNGs.

PRNGs are sometimes described as being *blocking* or *non-blocking*. For example, the Linux kernel PRNG provides two different RNGs, one of each type. A blocking RNG will prevent outputs

from the RNG from being delivered to the application requesting random numbers if it deems that doing so would be inappropriate for some reason.

6.2.2 Architectural model for PRNGs

An important basic architectural choice that is followed by most modern PRNGs is to separate the problems of entropy collection and generation of seeds from the problem of generating pseudo-random outputs as a function of the seed and generator state. NIST [264] provides a general functional model for describing and classifying PRNGs which makes this distinction clear. The components of this model include:

- **Entropy input**: this is provided to the PRNG for the purposes of generating the seed. This input is not guaranteed to be uniformly random, but is assumed to contain enough entropy that a seed of suitable quality can be extracted from it. This input must remain secret in order for the outputs of the PRNG to remain secure. This entropy input may initially be supplied by the user running the PRNG or may be harvested from the platform on which the PRNG is running.
- **Other inputs**: these might be time-based or take the form of a nonce. These inputs are not assumed to be secret. They are combined with the entropy input when generating seeds.
- **Personalisation string**: a further input to the seed generation process which is intended to provide further diversity for the generator outputs. For example, one might use different strings for key generation for different algorithms here.
- **Internal state:** this represents the memory of the PRNG, including the data that is used as input (and possibly modified) during the generation of outputs. Clearly, this state must remain secret for the future outputs of the PRNG to remain secure.
- **Instantiate function**: this function acquires the entropy input, any other input and the personalisation string and combines them to create a seed from which the initial internal state is created.
- **Generate function**: this function uses the current internal state to generate pseudo-random output bits and to update the state for the next request for output bits. This function should maintain a counter indicating the number of requests serviced or blocks of output produced since the generator was first seeded or reseeded. This counter would enable the PRNG to block further requests once a preset limit on the amount of output produced has been reached.
- **Reseed function**: this function combines a new entropy input (and possibly further additional input) with the current internal state to create a new seed and a new internal state.
- **Uninstantiate function**: this function erases the internal state; its intended use is to ensure the safe decommissioning of a PRNG.

- **Health test function**: this function is intended to provide a mechanism by which the PRNG can be tested to be functioning correctly.

We note that the last two components are often not explicitly present in PRNG implementations. Moreover, many PRNGs do not have "other inputs" or allow the use of personalisation strings. Some generators in the literature do not fully separate the reseed and generate functions, mixing entropy directly into the state of the generator, for example.

6.2.3 Security Requirements for PRNGs

Until quite recently, formal security requirements for PRNGs were lacking, and the requirements were informally stated and driven by the security requirements of the applications in which their outputs are intended to be used. The informal requirements can be stated as follows:

- **Output indistinguishability**: Without knowledge of the initial seed or current state, it should be hard to distinguish the outputs of the generator from a truly random sequence of the same type, even when many previous outputs are known. For certain generators, this property can be proven based on some computational assumption (e.g. the outputs of the Blum-Blum-Shub generator [50] are pseudo-random assuming the hardness of the quadratic residuosity problem, which is closely related to the factoring problem). For fast, practical generators, built using hash functions and block ciphers, this property rests on unproven but reasonable security assumptions concerning these symmetric components (e.g. the NIST CTR PRNG from [264] has output indistinguishability that relies on the block cipher acting as a pseudo-random function).

- **Forward security**: Compromise of the internal state of the generator should not allow an attacker to compute previous outputs of the generator, nor to distinguish previous outputs from random. This requirement implies in particular that it must be hard to compute any previous state of the generator from its current state. In turn, this implies that the generator state must be updated after each output in a one-way manner.

- **Resistance to state-extension attacks**: In a state-extension attack [193], an attacker is assumed to compromise the state of the generator, and then try to learn future outputs of the generator (or distinguish them from random). Clearly, in the absence of reseeding, this is possible since the future states and outputs are then a deterministic function of the current state. Moreover, if the reseeding process is carried out, but has insufficient entropy in its input, then an attacker can try to calculate forwards through the reseeding process, trying all likely values for the unknown entropy inputs used during the reseeding, and testing for consistency with some known outputs. It is desirable that a PRNG should resist such attack, since the design intention of reseeding is that it should assist in recovering from state compromises.

- **Compromise of reseeding data should not lead to generator compromise**: In some attack scenarios, the entropy input used during reseeding may fail to have insufficient entropy, or become known to the attacker. In this situation, we would like to ensure that the attacker cannot learn the generator's new state after reseeding, nor predict its outputs after reseeding. For this to be achieved, the entropy input must be carefully combined with the current state during reseeding.

Note that none of these requirements directly refer to the quality of entropy inputs, but that this rapidly emerges as a key concern in meeting the requirements.

6.2.4 Theoretical models

Theoretical models for the analysis of PRNGs first emerged in [23] and were significantly developed in [97, 98]. Generators secure in the models presented in these papers provably provide all of the above informally-stated security properties. The differ considerably in the way that they treat the incorporation of new entropy in the reseeding step. Generators in these models also deviate from the NIST architectural view discussed in Section 6.2.2, in that they do not consider other inputs, personalisation strings, the uninstantiate function, or the health test function. They all suffer from the unnatural requirement of having a random seed for an extractor (which may be known to the adversary) as part of the public parameters of the generator. This can be avoided in practice by replacing the seeded extractor with a concrete hash function.

These sources [23,97,98] have in common with [264] that they deliberately separate the concern of randomness generation for seeding/reseeding from the question of designing a generator taking assumed-to-be-random seeds/reseeds as input. Indeed, there are many good designs solving the latter problem, but few general-purpose solutions to the former.

6.2.5 Implementation considerations

In addition to meeting the above security requirements, there are many implementation issues that need to be addressed when deploying a PRNG.

Entropy sources: Foremost amongst these implementation issues are the questions of how to identify suitable sources of entropy, how to manage and process these sources, and how (and indeed, whether) to assess the quality of the entropy that is extracted from these sources when reseeding a PRNG. A good general overview of these issues can be found in [103, 137].

Entropy estimation: There is some debate in the literature on whether an implementation should try to estimate how much entropy is available from these sources. Accurate (or at least, conservative) estimation of entropy is important because of state extension attacks: too little entropy, and a state compromise (or a default initial state) can lead to predictable generator

outputs; on the other hand, waiting to long provides poor protection against state compromises, weakening forward security. The majority of practical PRNG designs do some form of entropy estimation. However, Ferguson *et al.* [123] contend that no procedure can accurately assess entropy (or rather, the amount of entropy unknown to an attacker) across all environments. Their `Fortuna` PRNG design attempts to get around the problem of entropy estimation by allocating gathered entropy, represented by events, to a sequence of entropy pools in order. The `Fortuna` generator then uses the pools at different intervals to reseed the generator. An analysis of this approach was recently provided in [98].

The `Fortuna` design sets out to avoid the need for entropy estimation whilst preventing state-extension attacks. As pointed out by Barak and Halevi [23], this approach works well so long as the entropy is well-spread across the different pools, but does not work well if the entropy is concentrated in one pool that is not often accessed when doing state refreshes. It is possible that an adversary could arrange for this to occur by generating large numbers of spurious events under his control. The view of Barak and Halevi is that it is better to accumulate entropy over a long period of time in a single pool and do infrequent reseeds, but without doing any entropy estimation, since in their view "*at best the entropy estimator provides a false sense of security*". A third approach is to perform conservative entropy estimation, and to reseed only when sufficient entropy is available – this is the approach taken in the Linux `dev/urandom` and `dev/random` PRNGs, for example.

Generator initialisation: An important special case of seeding is the setting of the initial state (which is done via the Instantiate function in the NIST model). A PRNG should be blocking until properly initialised, either with entropy supplied by the user, with entropy gathered from the local environment. There is anecdotal evidence that this is not popular with software developers – see [150], where it is explained how one SSH implementation uses the non-blocking Linux `dev/urandom` PRNG in preference to the blocking `dev/random` one when generating cryptographic keys. We reiterate that accessing a PRNG before it is properly seeded for the first time has been identified as a source of serious security problems, particularly in key generation [150].

6.2.6 Specific PRNGs and their analyses

Library functions in programming languages such as `random()` in the C programming language must be avoided in cryptographic applications. In general, such functions tend to be based on very weak generators such as Linear Congruential Generators. Dedicated cryptographic PRNG implementations are needed.

There are many operating-system-specific PRNGs. The Microsoft Windows PRNG has a closed-source implementation. An instance of this PRNG was reverse-engineered and found to have quite severe deficiencies in [99]. The Linux PRNG was analysed in [138], with significant attacks being found. The Linux PRNG has been modified as a result, and the current version was analysed in [209], with the previously reported weaknesses being found to have been largely addressed.

There are several PRNGs that are supplied as part of crypto libraries. Prominent amongst these is the OpenSSL PRNG. This generator has a rather ad hoc design. It was analysed in [306], and some changes were made as a result of this analysis. However, as far as we are aware, it has not been subjected to any further cryptographic analysis since then. Gutmann has designed a PRNG that is made available as part of his `cryptlib` software development kit[1]. This PRNG and its design are described in detail in [137].

The NIST special publication [264] contains several PRNG designs. As far as we are aware, none of these has been thoroughly analysed with the exception of the Dual Elliptic Curve generator. A pseudo-randomness property was proven for this generator in [62], based on some reasonable number-theoretic assumptions. However, the generator is relatively slow and known to have a small bias in its outputs. The generator has the potential to contain a backdoor, enabling its internal state to be reconstructed given sufficient output [327], and it is widely believed that this potential was exploited during the NIST standardisation process by NSA. A recent study [71] found the generator to be in surprisingly widespread use. The controversy surrounding this Dual Elliptic Curve generator led to the withdrawal of the generator from the NIST special publication [264] and the opening of a comment period on a revised version of the NIST document[2].

These NIST PRNG designs do not include a full specification of how to gather and process entropy sources for seeding/reseeding purposes, which is consistent with the over-arching approach in [264].

The `Fortuna` generator from [123] incorporates learning from the earlier `Yarrow` design [192]. It's basic design of using entropy pools to collect entropy for reseeding at different rates was recently validated by the analysis of [98], whilst see [142, 326] for two analyses of Intel's hardware RNG.

6.2.7 Designing around bad randomness

Given that randomness failures seem to be hard to avoid in general, a number of authors have attempted to design cryptosystems that handle bad randomness to the extent that this is possible. Work in this direction can be summarised as follows:

- For signatures, there is a folklore de-randomisation technique which neatly sidesteps security issues arising from randomness failures: simply augment the signature scheme's private key with a key for a pseudo-random function (PRF), and derive any randomness needed during signing by applying this PRF to the message to be signed; meanwhile verification proceeds as normal.

- In the symmetric encryption setting, Rogaway [298] argued for the use of nonce-based encryption, thus reducing reliance on randomness. Rogaway and Shrimpton [302] initiated the study of misuse-resistant authenticated encryption (AE), considering the residual security of

[1] See `http://www.cryptlib.com/`

[2] See `http://www.nist.gov/itl/csd/sp800-90-042114.cfm`.

AE schemes when nonces are repeated. Katz and Kamara [187] considered the security of symmetric encryption in a chosen-randomness setting, wherein the adversary has complete control over the randomness used for encryption (except for the challenge encryption which uses fresh randomness).

- In the public key encryption (PKE) setting, Bellare *et al.* [29] considered security under *chosen distribution attack*, wherein the joint distribution of message and randomness is specified by the adversary, subject to containing a reasonable amount of min entropy. Bellare *et al.* gave several designs for PKE schemes achieving this notion in the Random Oracle Model (ROM) and in the standard model. A follow-up work [291] considers a less restrictive adversarial setting.

- Also in the PKE setting, Yilek [355], inspired by virtual machine reset attacks in [293], considered the scenario where the adversary can force the reuse of random values that are otherwise well-distributed and unknown to the adversary. This is referred to in [355] as the *Reset Attack* (RA) setting. In [355], Yilek also gave a general construction achieving security for public key encryption in his RA setting. The RA setting was recently extended to a setting where the adversary can to a certain extent control the randomness that is used during encryption, the so-called Related Randomness Attack (RRA) setting [270].

- Ristenpart and Yilek [293] studied the use of "hedging" as a general technique for protecting against broad classes of randomness failures in already-deployed systems, and implemented and benchmarked this technique in OpenSSL. Hedging in the sense of [293] involves replacing the random value r required in some cryptographic scheme with a hash of r together with other contextual information, such as a message, algorithm or unique operation identifier, etc. Their results apply to a variety of different randomness failure types but have their security analyses restricted to the ROM.

6.3 Key Life Cycle Management

In this section we discuss general aspects related to key life cycle management. More information about key management techniques can be found in [269][Chapter 13], and in NIST-800-57 [262].

Objectives of Key Management

Cryptographic mechanisms reduce the problem of data security to the problem of key management. This is known as Kerckhoffs' principle: *the security of a cryptosystem should not rely on the secrecy of any of its workings, except for the value of the secret key.* It follows that good key management is essential in order to benefit from the introduction of cryptography. We distinguish the following objectives of key management:

1. Protecting the confidentiality and authenticity of secret and private keys, as well as protecting secret and private keys against unauthorised use.
2. Protecting the authenticity of public keys.
3. Ensuring the availability of secret and public keys.

To accomplish these three goals we need to examine the whole key life cycle; from generation of key the material through to destruction.

Key Generation

Secret keys and private keys need to be unpredictable. Symmetric primitives usually don't have additional requirements for the secret keys, except that some primitives have a small fraction of weak keys, which should not be used. Asymmetric primitives usually have additional requirements, both on their private and public keys. For example, they often require the generation of prime numbers that need to satisfy extra properties. Keys can either be generated at random in a protocol, in which case generating them with a sufficient amount of entropy turns out to be a very challenging task in practice, see Section 6.2, in other instances keys are derived from other data as part of the protocol definition. There are numerous well documented attacks on systems for which not enough entropy was used to generate the underlying key material.

Key Registration/Certification

Keys need to be associated with their owner (user). For example, public keys are linked to their owner by means of (public-key) certificates. Through the issuing of a certificate, a certification authority guarantees that a certain key belongs to a certain user, and associated policy statements specify for what purposes the owner may use the key. A certificate also has a validity period. Certificates are usually public documents. Their authenticity is ensured by means of a digital signature, placed by the certification authority. However, one needs to trust the certificate authority and its public key, which is itself authenticated by another certificate authority; creating a certificate chain. At the root of the chain is a root certificate authority. These root certificates can be distributed to relying parties and signatories alike by, for example including them in applications (as in a web browser) or having them downloaded from an authoritative source (e.g. a designated public authority), for the purpose of invoking trust.

Various issues have come to light in the last few years as to the ability for users to fully trust the root certificates in their browsers. Thus certification is a technology which is (still) not completely 100 percent reliable. Hence, when using certificates in a non-public application (e.g. in a corporate environment) care needs to be taken as to the underlying policy framework and how this is implemented and enforced.

Key Distribution and Installation

Keys need to be distributed to their users. For systems based on symmetric cryptography, both the sender and the receiver need to obtain a copy of the key, hence the key needs to be transported securely (protection of confidentiality and authenticity) at least once, or agreed via means of a key agreement scheme. All the copies of the key need to be installed and stored securely. For systems based on asymmetric cryptography, the private key is often generated where it will be used, such that no transport is needed. In secure hardware, the functionality to export the private key of an asymmetric key pair is usually deliberately not implemented (again, this can be compromised by poorly implemented hardware in some cases). Otherwise, it needs to be protected like a symmetric key. The public key still needs to be transported, but only the authenticity (and hence the integrity) needs to be protected, which is achieved by the use of certificates.

In order to reduce the number of keys that need to be stored locally, one can use Key Distribution Centers, centrally managed key servers. Users share long-term keys with the Key Distribution Centers and trust the servers to provide them with the keys of the other users when they need them. Key Distribution Centers can manage both secret and public keys.

Key Use

The goal of key management is to put keys in place such that they can be used for a certain period of time. During the lifetime of a key, it has to be protected against unauthorised use by attackers. The key must also be protected against unauthorised uses by the owner of the key, e.g. even the owner of the key should not be allowed to export a key or to use it in an insecure environment. This protection can be provided by storing the key on secure hardware and by using secure software, which includes authorisation checks.

Key Storage

By using secure hardware, it is possible to store keys such that they can never be exported, and hence are very secure against theft or unauthorised use. However, sometimes keys get lost and it might be desirable to have a backup copy. Organisations might require backups of keys in order to be able to access data after employees leave. Similarly, expired keys might be archived in order to keep old data accessible. Finally, under certain conditions law enforcement agencies might request access to certain keys. Technical systems that implement access for law enforcement agencies are called key escrow mechanisms or key recovery mechanisms.

Backup, archival and escrow/recovery of keys complicate key management, because they increase the risk for loopholes for unauthorised access to keys. The advanced security requirement of non-repudiation requires that the owner of a key is the only one who has access to the key at all times from generation to key retirement. For example, keys that are used for advanced electronic signatures have to be under the sole control of the user. Archival, backup or storage of such keys

is difficult. For use of the non-repudiation property in a court of law one may require special procedures for digital signature generation to be followed.

Revocation/Validation

Cryptographic keys expire and are replaced. Sometimes it can happen that keys have to be taken out of use before the planned end of their lifetime, e.g. if secret keys leak to outsiders or if developments in cryptanalysis make schemes insecure. This process is called revocation. In centralised systems, revocation can usually be achieved relatively easily, but in distributed systems special measures have to be implemented to avoid that people use or rely on keys that have been expired early. In the context of revocation, validation has a very specific meaning. It means to check whether a cryptographic operation, e.g. placing a digital signature, was performed with a key that is valid, or was valid at the time the operation took place.

Key Archive/Destruction

When the lifetime of the key has expired, it has to be removed from the hardware. This requires a secure deletion process. In most operating systems and applications, the deletion of a file only clears a logic flag. It doesn't result in actual removal of the data until the disk space used to store the file is reclaimed and overwritten by another application. On many file storage media, even after a file has been overwritten, it is possible to recover the original file, using some moderately advanced equipment. This is called data remanence. Various techniques have been developed to counter data remanence. At the logical level, one can overwrite the disk space repeatedly with certain bit patterns in order to make recovery difficult. At the physical level, one can degauss (on magnetic media) or employ other operations that restore the storage media in pristine state, or one can physically destroy the storage media.

6.3.1 Key Management Systems

In many large organisations there is a need to systematise the above mentioned aspects of key life-cycle. This is usually done using a *Cryptographic Key Management System*; this is an automated system consisting of hardware and software components which implement the required policy to manage the above keys. Aspects including generation, storage, validation and use. For example if keys are held in hardware security modules, then it is common practice to only enable extraction of keys from the hardware modulus under some form of key wrap algorithm. A cryptographic key management system ensures that such a policy is enforced, without the users being able to override it.

The NIST standard 800-130 [252] provides a framework for describing such key management systems. In a way which enables a simpler validation that any specific key management system satisfies the given policy. The framework defines specific topics and for each topic defines a set of

requirements which any framework needs to meet, from this any given system can be mapped onto the framework by stating how and in what way the specific system meets the given framework.

Bibliography

[1] Masayuki Abe, editor. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*. Springer, 2010.

[2] 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.

[3] 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.

[4] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *IJWMC*, 5(1):48–59, 2011.

[5] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of des and aes, secure against some attacks. In Çetin Kaya Koç et al. [69], pages 309–318.

[6] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the security of RC4 in TLS. In Samuel T. King, editor, *USENIX Security*, pages 305–320. USENIX Association, 2013.

[7] 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.

[8] 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.

[9] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Knudsen [198], pages 83–107.

[10] ANSI X9.102. Symmetric key cryptography for the financial services industry - wrapping of keys and associated data. American National Standard Institute, 2008.

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

[12] ANSI X9.24. Retail financial services symmetric key management part 1: Using symmetric techniques. American National Standard Institute, 2009.

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

[14] 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.

[15] ANSI X9.82. Random number generation part 1: Overview and basic principles. American National Standard Institute, 2006.

[16] 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.

[17] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In Matsui [226], pages 578–597.

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

[19] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy*, pages 3–11. IEEE Computer Society, 2004.

[20] 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.

[21] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of Salsa, ChaCha, and Rumba. In Nyberg [265], pages 470–488.

[22] Steve Babbage and Matthew Dodd. The mickey stream ciphers. In Robshaw and Billet [295], pages 191–209.

[23] Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 203–212. ACM, 2005.

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

[25] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Safavi-Naini and Canetti [308], pages 608–625.

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

[27] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *POPL*, pages 90–101. ACM, 2009.

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

[29] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Matsui [226], pages 232–249.

[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 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.

[32] 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.

[33] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Roy and Meier [305], pages 389–407.

[34] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a fast software-oriented stream cipher. In Robshaw and Billet [295], pages 98–118.

[35] Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. In Robshaw [294], pages 15–29.

[36] Daniel J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2005.

[37] Daniel J. Bernstein. Snuffle 2005: the Salsa20 encryption function, 2007. http://http://cr.yp.to/snuffle.html.

[38] Daniel J. Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko van Someren. Factoring rsa keys from certified smart cards: Coppersmith in the wild. In Sako and Sarkar [309], pages 341–360.

[39] Eli Biham. A fast new des implementation in software. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1997.

[40] Eli Biham and Yaniv Carmeli. Efficient reconstruction of RC4 keys from internal states. In Nyberg [265], pages 270–288.

[41] Eli Biham, Orr Dunkelman, and Nathan Keller. A related-key rectangle attack on the full KASUMI. In Roy [304], 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 Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 299–319. Springer, 2010.

[45] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In Matsui [226], pages 1–18.

[46] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs 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] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In Wiener [351], pages 216–233.

[48] 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.

[49] 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.

[50] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[51] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Lee and Wang [212], pages 344–371.

[52] 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.

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

[54] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Kilian [194], pages 213–229.

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

[56] 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.

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

[58] 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.

[59] Ernest F. Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2000.

[60] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009.

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

[62] Daniel R. L. Brown and Kristian Gjøsteen. A security analysis of the nist sp 800-90 elliptic curve random number generator. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2007.

[63] 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.

[64] 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.

[65] Mihir Bellare Ran Canetti and Hugo Krawczyk. Keying hash functions for message authentication. In Koblitz [199], pages 1–15.

[66] Christophe De Cannière and Christian Rechberger. Preimages for reduced SHA-0 and SHA-1. In Wagner [345], pages 179–202.

[67] 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.

[68] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[69] Çetin Kaya Koç, David Naccache, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*. Springer, 2001.

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

[71] Stephen Checkoway, Matthew Fredrikson, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, and Hovav Shacham. On the practical exploitability of Dual EC in TLS implementations. In *USENIX Security Symposium*, 2014.

[72] 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.

[73] 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.

[74] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Vaudenay [343], pages 1–11.

[75] Olivier Chevassut, Pierre-Alain Fouque, Pierrick Gaudry, and David Pointcheval. Key derivation and randomness extraction. *IACR Cryptology ePrint Archive*, 2005:61, 2005.

[76] Joo Yeon Cho and Miia Hermelin. Improved linear cryptanalysis of sosemanuk. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 101–117. Springer, 2009.

[77] Carlos Cid and Gaëtan Leurent. An analysis of the XSL algorithm. In Roy [304], pages 333–352.

[78] Don Coppersmith. Finding a small root of a bivariate integer equation; Factoring with high bits known. In Maurer [227], pages 178–189.

[79] Don Coppersmith. Finding a small root of a univariate modular equation. In Maurer [227], pages 155–165.

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

[81] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, and Michael K. Reiter. Low-exponent RSA with related messages. In Maurer [227], pages 1–9.

[82] 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.

[83] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Knudsen [198], pages 272–287.

[84] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New attacks on PKCS#1 v1.5 encryption. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 369–381. Springer, 2000.

[85] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On the security of RSA padding. In Wiener [351], pages 1–18.

[86] 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.

[87] 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.

[88] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

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

[90] Ivan Damgård. A design principle for hash functions. In Brassard [58], pages 416–427.

[91] Nasser Ramazani Darmian. A distinguish attack on rabbit stream cipher based on multiple cube tester. *IACR Cryptology ePrint Archive*, 2013:780, 2013.

[92] Debian. Debian Security Advisory DSA-1571-1: OpenSSL – predictable random number generator, 2008. `http://www.debian.org/security/2008/dsa-1571`.

[93] 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.

[94] 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.

[95] Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *J. Cryptology*, 27(2):183–209, 2014.

[96] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Practical complexity cube attacks on round-reduced keccak sponge function. *IACR Cryptology ePrint Archive*, 2014:13, 2014.

[97] Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 647–658. ACM, 2013.

[98] Yevgeniy Dodis, Adi Shamir, Noah Stephens-Davidowitz, and Daniel Wichs. How to eat your entropy and have it too - optimal recovery strategies for compromised rngs. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO (2)*, volume 8617 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2014.

[99] Leo Dorrendorf, Zvi Gutterman, and Benny Pinkas. Cryptanalysis of the random number generator of the Windows operating system. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.

[100] 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.

[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 3rd, J. Schiller, and S. Crocker. Randomness Requirements for Security. RFC 4086 (Best Current Practice), June 2005.

[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] 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.

[113] ENISA. Algorithms, key size and parameters report – 2013 recommendations. ENISA XXXX, 2013.

[114] ENISA. Protocols report – 2014 recommendations. ENISA XXXX, 2014.

[115] Matthias Ernst, Ellen Jochemsz, Alexander May, and Benne de Weger. Partial key exposure attacks on RSA up to full size exponents. In Cramer [88], pages 371–386.

[116] 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.

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

[118] 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`.

[119] European Payments Council. Guidelines on algorithms usage and key management, 2013.

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

[121] Federal Information Processing Standards Publication 202. SHA-3 standard: Permutation-based hash and extendable-output functions (draft). National Institute of Standards and Technology, 2014.

[122] Xiutao Feng, Jun Liu, Zhaocun Zhou, Chuankun Wu, and Dengguo Feng. A byte-based guess and determine attack on sosemanuk. In Abe [1], pages 146–157.

[123] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering — Design Principles and Practical Applications*. Wiley, 2010.

[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] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Kilian [194], pages 260–274.

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

[128] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç et al. [69], pages 251–261.

[129] 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.

[130] 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.

[131] Ian Goldberg and David Wagner. Randomness and the Netscape browser, 1996. `http://www.drdobbs.com/windows/184409807`.

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

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

[134] Louis Goubin and Ange Martinelli. Protecting aes with shamir's secret sharing scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2011.

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

[136] 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 Abe [1], pages 56–75.

[137] Peter Gutmann. Software generation of practically strong random numbers. In Aviel D. Rubin, editor, *USENIX Security*. USENIX Association, 1998.

[138] Zvi Gutterman, Benny Pinkas, and Tzachy Reinman. Analysis of the linux random number generator. In *IEEE Symposium on Security and Privacy*, pages 371–385. IEEE Computer Society, 2006.

[139] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Canteaut and Viswanathan [67], 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 [267], pages 292–304.

[142] Mike Hamburg, Paul Kocher, and Mark E. Marson. Analysis of Intel's Ivy Bridge digital random number generator, March 2012. `http://www.cryptography.com/public/pdf/Intel_TRNG_Report_20120312.pdf`.

[143] Helena Handschuh and Bart Preneel. Key-recovery attacks on universal hash function based MAC algorithms. In Wagner [345], pages 144–161.

[144] D. Harkins. Synthethic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). RFC 5297 (Informational), October 2008.

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

[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] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. The Grain family of stream ciphers. In Robshaw and Billet [295], pages 179–190.

[149] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *IJWMC*, 2(1):86–93, 2007.

[150] 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.

[151] 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.

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

[153] R. Housley and M. Dworkin. Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm. RFC 5649 (Informational), August 2009.

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

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

[156] 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.

[157] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[158] Takanori Isobe, Toshihiro Ohigashi, Yuhei Watanabe, and Masakatu Morii. Full plaintext recovery attack on broadcast rc4. In Moriai [239], pages 179–202.

[159] ISO/IEC 10118-2:2010. Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher. International Organization for Standardization, 2010.

[160] ISO/IEC 11770-6. Information technology – Security techniques – Key management – Part 6: Key derivation. International Organization for Standardization, Under Development.

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

[162] 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.

[163] ISO/IEC 18031. Information technology – Security techniques – Random bit generator. International Organization for Standardization, 2011.

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

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

[166] ISO/IEC 19772. Information technology – Security techniques – authenticated encryption. International Organization for Standardization, 2009.

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

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

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

[170] 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.

[171] 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.

[172] 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.

[173] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In Safavi-Naini and Canetti [308], pages 31–49.

[174] 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.

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

[176] 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.

[177] 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`.

[178] 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`.

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

[180] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In Johansson and Nguyen [174], pages 177–193.

[181] Antoine Joux. A new index calculus algorithm with complexity $$l(1/4+o(1))$$ in small characteristic. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 355–379. Springer, 2013.

[182] 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.

[183] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002.

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

[185] 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.

[186] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898 (Informational), September 2000.

[187] Seny Kamara and Jonathan Katz. How to encrypt with a malicious random number generator. In Nyberg [265], pages 303–315.

[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] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant aes-gcm. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009.

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

[192] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In Howard M. Heys

and Carlisle M. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 13–33. Springer, 1999.

[193] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *FSE*, volume 1372 of *Lecture Notes in Computer Science*, pages 168–188. Springer, 1998.

[194] 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.

[195] A. Kircanski and A. M. Youssef. On the sliding property of SNOW 3G and SNOW 2.0. *IET Inf. Secur.*, 5(4):199–206, 2011.

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

[197] 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.

[198] 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.

[199] Neal Koblitz, editor. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996.

[200] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz [199], pages 104–113.

[201] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [351], pages 388–397.

[202] Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A high-performance conventional authenticated encryption mode. In Roy and Meier [305], pages 408–426.

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

[204] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In Kilian [194], pages 310–331.

[205] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Rabin [290], pages 631–648.

[206] Hugo Krawczyk. Hmac-based extract-and-expand key derivation function (hkdf). RFC 5869 (Informational), 2010.

[207] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the tls protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2013.

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

[209] Patrick Lacharme, Andrea Röck, Vincent Strubel, and Marion Videau. The linux pseudorandom number generator revisited. *IACR Cryptology ePrint Archive*, 2012:251, 2012.

[210] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In Matsui [226], pages 126–143.

[211] Franck Landelle and Thomas Peyrin. Cryptanalysis of full RIPEMD-128. In Johansson and Nguyen [174], pages 228–244.

[212] 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.

[213] Jung-Keun Lee, Dong Hoon Lee, and Sangwoo Park. Cryptanalysis of sosemanuk and snow 2.0 using linear masks. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 524–538. Springer, 2008.

[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. Public keys. In Safavi-Naini and Canetti [308], pages 626–642.

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

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

[220] 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`.

[221] Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In Shoup [325], pages 97–117.

[222] Subhamoy Maitra and Goutam Paul. New form of permutation bias and secret key leakage in keystream bytes of RC4. In Nyberg [265], pages 253–269.

[223] James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1 v2.0. In Kilian [194], pages 230–238.

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

[225] 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.

[226] 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.

[227] 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.

[228] Alexander Maximov and Alex Biryukov. Two trivial attacks on Trivium. In Adams et al. [2], pages 36–55.

[229] Alexander Maximov and Dmitry Khovratovich. New state recovery attack on RC4. In Wagner [345], pages 297–316.

[230] 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.

[231] David A. McGrew and John Viega. The security and performance of the Galois/Counter mode (GCM) of operation. In Canteaut and Viswanathan [67], pages 343–355.

[232] 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.

[233] Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced SHA-256. In Johansson and Nguyen [174], pages 262–278.

[234] Florian Mendel, Thomas Peyrin, Martin Schläffer, Lei Wang, and Shuang Wu. Improved cryptanalysis of reduced ripemd-160. In Sako and Sarkar [309], pages 484–503.

[235] 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.

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

[237] 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.

[238] Ralph C. Merkle. A certified digital signature. In Brassard [58], pages 218–238.

[239] Shiho Moriai, editor. *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*. Springer, 2014.

[240] 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.

[241] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014.

[242] 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.

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

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

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

[246] 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.

[247] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[248] NIST Special Publication 180-4. Secure hash standard (SHS). National Institute of Standards and Technology, 2012.

[249] NIST Special Publication 186-4. Digital signature standard (DSS). National Institute of Standards and Technology, 2013.

[250] NIST Special Publication 198-1. The keyed-hash message authentication code (HMAC). National Institute of Standards and Technology, 2008.

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

[252] NIST Special Publication 800-130. A framework for designing cryptographic key management systems. National Institute of Standards and Technology, 2013.

[253] NIST Special Publication 800-132. Recommendation for password-based key derivation – Part 1: Storage applications. National Institute of Standards and Technology, 2010.

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

[255] 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.

[256] 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.

[257] 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.

[258] NIST Special Publication 800-38F. Recommendation for block cipher modes of operation – Methods for Key Wrapping. National Institute of Standards and Technology, 2012.

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

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

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

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

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

[264] NIST Special Publication 800-90A. Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology, 2012.

[265] 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.

[266] Kaisa Nyberg and Johan Wallén. Improved linear distinguishers for SNOW 2.0. In Robshaw [294], pages 144–162.

[267] 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.

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

[269] Christof Paar and J.XXXX Pelzl. *Understanding cryptography: A textbook for students and practitioners*. Springer, 2009.

[270] Kenneth G. Paterson, Jacob C. N. Schuldt, and Dale L. Sibborn. Related randomness attacks for public key encryption. In Hugo Krawczyk, editor, *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2014.

[271] 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 [212], pages 161–178.

[272] Kenneth G. Paterson and Arnold K. L. Yau. Padding Oracle Attacks on the ISO CBC Mode Encryption Standard. In Okamoto [267], pages 305–323.

[273] C. Percival and S. Josefsson. The scrypt Password-Based Key Derivation Function draft-josefsson-scrypt-kdf-01. Internet-Draft (Informational), September 2012.

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

[275] Raphael Chung-Wei Phan. Related-key attacks on triple-DES and DESX variants. In Okamoto [267], pages 15–24.

[276] 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.

[277] Leon A. Pintsov and Scott A. Vanstone. Postal revenue collection in the digital age. In Yair Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, volume 1962 of *Lecture Notes in Computer Science*, pages 105–120. Springer, 2000.

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

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

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

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

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

[283] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.

[284] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2, 300 ge. *J. Cryptology*, 24(2):322–345, 2011.

[285] 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.

[286] 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.

[287] Gordon Procter and Carlos Cid. On weak keys and forgery attacks against polynomial-based mac schemes. In Moriai [239], pages 287–304.

[288] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Johansson and Nguyen [174], pages 142–159.

[289] Niels Provos and David Mazières. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91. USENIX, 1999.

[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] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Johansson and Nguyen [174], pages 93–110.

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

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

[294] Matthew J. B. Robshaw, editor. *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*. Springer, 2006.

[295] Matthew J. B. Robshaw and Olivier Billet, editors. *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer, 2008.

[296] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 98–107. ACM, 2002.

[297] 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.

[298] Phillip Rogaway. Nonce-based symmetric encryption. In Roy and Meier [305], pages 348–359.

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

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

[301] 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.

[302] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Vaudenay [343], pages 373–390.

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

[304] 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.

[305] 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.

[306] Markku-Juhani O. Saarinen. Weakness of the openssl prng in versions up to openssl 0.9.6a, 2001. `http://mjos.fi/doc/secadv_prng.txt`.

[307] 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.

[308] 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.

[309] Kazue Sako and Palash Sarkar, editors. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*. Springer, 2013.

[310] 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.

[311] 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.

[312] 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.

[313] 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.

[314] Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on whirlpool: Improved preimage and collision attacks. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 562–579. Springer, 2012.

[315] 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.

[316] J. Schaad and R. Housley. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394 (Informational), September 2002.

[317] 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.

[318] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Brassard [58], pages 239–252.

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

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

[321] 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.

[322] Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux. Statistical attack on RC4 - distinguishing WPA. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 343–363. Springer, 2011.

[323] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

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

[325] 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.

[326] Thomas Shrimpton and R. Seth Terashima. A provable security analysis of intel's secure key rng. *IACR Cryptology ePrint Archive*, 2014:504, 2014.

[327] Dan Shumov and Nils Ferguson. On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng, August 2007. Rump session presentation at Crypto 2007, `http://rump2007.cr.yp.to/15-shumow.pdf`.

[328] Sergei P. Skorobogatov. Using optical emission analysis for estimating contribution to power analysis. In Luca Breveglieri, Israel Koren, David Naccache, Elisabeth Oswald, and Jean-Pierre Seifert, editors, *FDTC*, pages 111–119. IEEE Computer Society, 2009.

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

[330] Marc Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In Johansson and Nguyen [174], pages 245–261.

[331] 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.

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

[333] 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.

[334] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.

[335] Elena Trichina, Tymur Korkishko, and Kyung-Hee Lee. Small size, low power, side channel-immune aes coprocessor: Design and synthesis results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2004.

[336] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on aes, and countermeasures. *J. Cryptology*, 23(1):37–71, 2010.

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

[338] 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.

[339] Paul C. van Oorschot and Michael J. Wiener. A known plaintext attack on two-key triple encryption. In Ivan Damgård, editor, *EUROCRYPT*, volume 473 of *Lecture Notes in Computer Science*, pages 318–325. Springer, 1990.

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

[341] 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.

[342] Serge Vaudenay. Security flaws induced by CBC padding - Applications to SSL, IPSEC, WTLS ... In Knudsen [198], pages 534–546.

[343] 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.

[344] Serge Vaudenay and Martin Vuagnoux. Passive-only key recovery attacks on RC4. In Adams et al. [2], pages 344–359.

[345] 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.

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

[347] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Shoup [325], pages 17–36.

[348] Brent Waters. Efficient identity-based encryption without random oracles. In Cramer [88], pages 114–127.

[349] 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.

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

[351] 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.

[352] Hongjun Wu. A new stream cipher hc-256. In Roy and Meier [305], pages 226–244.

[353] Hongjun Wu. The stream cipher hc-128. In Robshaw and Billet [295], pages 39–47.

[354] 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.

[355] Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2010.

Index

Catalogue number TP-05-14-084-EN-N, ISBN 978-92-9204-102-1, DOI 10.2824/36822

ENISA
European Union Agency for Network and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office
1 Vassilis Sofias, Marousi 151 24, Athens, Greece

Catalogue number TP-05-14-084-EN-N
ISBN 978-92-9204-102-1
DOI 10.2824/36822

PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu