MINISTRY OF COMMUNICATIONS AND MULTIMEDIA MALAYSIA

CyberSecurity MALAYSIA

MySEAL Project:

Submission and Evaluation Criteria

Version 2.0 [2018]

Name of author: MySEAL FOCUS GROUP

File name: CD-5-RPT-0118-MySEAL Criteria Version 2.0-V1a

Date of document: 08 April 2018

Document classification : Public

For inquiry about this document please contact:
Hazlin Abdul Rani
Head, Cryptography Development Department
hazlin@cybersecurity.my

For general inquiry about us or our services,
please email: info@cybersecurity.my

Preface

This document is issued to provide information to the public regarding the submission and evaluation criteria for the selection of algorithms to be listed in *Senarai Algoritma Kriptografi Terpercaya Negara* (MySEAL) / National Trusted Cryptographic Algorithm List. MySEAL will be used as a requirement and guideline on the usage of cryptographic algorithms in all trusted cryptography products in Malaysia.

The document consists of 8 sections. Section 1 provides the introduction of MySEAL. Section 2, Section 3 and Section 4 explain the general requirements, submission criteria and evaluation criteria for each primitive respectively. Section 5 provides licensing requirements for each submission while Section 6 presents the timeline for this project. Section 7 explains the formal submission requirements, whereas Section 8 provide other general information regarding MySEAL project.

This document has been developed by MySEAL Focus Group members which consists of national cryptographic experts from public universities, private universities and government agencies. Any enquiries pertaining to this document can be submitted to myseal.fg@cybersecurity.my.

Acknowledgement

Special thanks and appreciation to the following organisations / institutions on the development of this document (in alphabetical order): -

a. CyberSecurity Malaysia
b. MIMOS Berhad
c. Universiti Kebangsaan Malaysia
d. Universiti Malaya
e. Universiti Malaysia Sabah
f. Universiti Multimedia
g. Universiti Pertahanan Nasional Malaysia
h. Universiti Putra Malaysia
i. Universiti Sains Islam Malaysia
j. Universiti Sains Malaysia
k. Universiti Teknikal Malaysia Melaka
l. Universiti Teknologi MARA
m. Universiti Tenaga Nasional
n. Universiti Tunku Abdul Rahman

Table of Contents

1.0 MySEAL Introduction

Senarai Algoritma Kriptografi Terpercaya Negara (MySEAL) is a project to develop a portfolio of national trusted cryptographic algorithms. It is a project specifically designed to provide a list of cryptographic algorithms suitable for implementation within Malaysian context that supports National Cryptography Policy (NCP). While NCP serves as a guiding document for Malaysia to achieve cryptographic sovereignty, MySEAL will support in the scientific areas of cryptography and cryptanalysis.

This exercise is to encourage strategic collaboration, research and production of cryptographic systems by local industry, where they can submit their cryptographic algorithms for testing and validation by examiners before the algorithms can be acknowledged as trusted cryptographic algorithms at national level. For a cryptographic algorithm to be listed into MySEAL, it needs to comply with the criteria as stated in this document. These criteria have been developed based on accepted international standards and requirements defined by MySEAL Focus Group committee. This committee is spearheaded by CyberSecurity Malaysia and supported by members from Malaysian institutions.

During the first round of MySEAL project, it accepts only symmetric (block cipher and stream cipher), asymmetric and cryptographic hash function primitives. After further consideration by MySEAL Focus Group committee, two more cryptographic primitives; cryptographic key generation primitive and cryptographic pseudo random number generator primitive are considered to be included in MySEAL project. The algorithms for each primitive will be obtained in two ways; a call for submission of new algorithms, and algorithms from existing standards including other cryptographic algorithm listing projects selected by an appointed committee. All algorithms will first be vetted and then be thoroughly evaluated based on the evaluation criteria and finally, selected algorithms will be announced.

MySEAL initiative is by no means a small feat. Ever since the documentation of the National IT Agenda (NITA) in 1996 which listed down e-Sovereignty as one of Malaysia’s objectives in entering the era of Information Technology (IT), the execution of MySEAL is a major milestone for Malaysia. It is through this initiative that Malaysia will enter into the realm of information security fundamentals. This challenging arena will attest Malaysia’s perseverance and stamina in protecting her information infrastructure at the cryptographic algorithm level.

Besides providing challenges and aspirations to Malaysian cryptographers, this project also aims at nurturing new talent and retaining existing talent. It is with this note that MySEAL initiative has given Malaysia a golden opportunity to provide a collaborative platform between government entities, industries and higher institutions, to promote and encourage participants in developing new cryptographic algorithms as well as producing new cryptographers. Thus, MySEAL project will bring Malaysia a step closer to realize Malaysia's Vision 2020 sixth challenge, that is *establishing a scientific and progressive society, a society that is innovative and forward-looking, one that is not only a consumer of technology but also a contributor to the scientific and technological civilisation of the future*.

2.0 MySEAL Requirements

This section presents MySEAL requirements; the categories of cryptographic primitives that are considered in MySEAL project, the eligibility of participants, the eligibility of cryptographic algorithms to be submitted and the project's general selection criteria required for all primitives. Security, efficiency and flexibility of cryptographic primitives are further discussed under the sub section of general selection criteria.

2.1 Categories of Cryptographic Primitives

MySEAL is seeking submissions of strong cryptographic primitives in the categories given below:

a) Block Cipher Primitives
b) Stream Cipher Primitive
c) Asymmetric Cryptographic Primitive
d) Cryptographic Hash Function Primitives
e) Cryptographic Key Generation Primitive
f) Cryptographic Pseudo Random Number Generator Primitive

2.2 Eligibility of Participants

Submission is open to the inventor and/or owner of the algorithm.

2.3 Eligibility of Cryptographic Algorithms

Any cryptographic algorithm that is not listed in any known standard and other cryptographic algorithm listing project.

2.4 General Selection Criteria

The main selection criteria consist of security, efficiency and flexibility.

1) **Security**

   It is mandatory to assess the fundamental cryptographic security level through cryptanalysis.

2) **Efficiency**

   It is essential for a cryptographic algorithm to be efficient on hardware and/or software deployment.

3) **Flexibility**

It is desirable for a cryptographic algorithm to be suitable for use in a wide-range of environments.

3.0 Submission Criteria for Each Primitive

This section provides information on the submission criteria for each primitive as in the following subsections.

3.1 Block Cipher Primitives

Block cipher primitives are divided into two categories; general-purpose block cipher and lightweight block cipher. The submission criteria are as follows:

1) Block Cipher

a. Key length of at least 128 bits.
b. Block length of at least 128 bits.
c. Security analysis report shall include but not limited to:
    i. NIST statistical tests
    ii. Linear cryptanalysis
    iii. Differential cryptanalysis
d. Implementation and performance reports:
    i. Targeted software and/or
    ii. Targeted hardware
e. Justification on design principles of the algorithm. See **Annex A** for an example.
f. Test vectors
    - Number of keys: - a minimum of 3 for each key size
    - Number of plaintext-ciphertext pairs: - 3 for each key
    - Processing sample must be in ECB mode with bit ‘0’ padding
    - Intermediate output for each round

2) Lightweight Block Cipher

a. Key length of at least 80 bits.
b. Block length of at least 64 bits.
c. Security analysis report shall include but not limited to:
    i. NIST statistical tests
    ii. Linear cryptanalysis
    iii. Differential cryptanalysis

d. Implementation and performance reports:
    i. Targeted software and/or
        a) Program code size
        b) RAM size
    ii. Targeted hardware
        a) Chip area
        b) Cycle
        c) Bits per cycle
        d) Power
        e) Energy

e. Justification on design principles of the algorithm. See **Annex B** for an example.

f. Test vectors
    - Number of keys: - a minimum of 3 for each key size
    - Number of plaintext-ciphertext pairs: - 3 for each key
    - Processing sample must be in ECB mode with bit ‘0’ padding
    - Intermediate output for each round

3.2 Stream Cipher Primitive

The submission criteria are as follows:

a. Operation: Synchronous/self-synchronous stream cipher

b. Hardware
    i. Key length of at least 80 bits.
    ii. Internal memory of at least 160 bits.

c. Software
    i. Key length of at least 128 bits.
    ii. Internal memory of at least 256 bits.

d. Security analysis report shall include but not limited to:
    i. NIST statistical tests
    ii. Algebraic attack
    iii. Correlation attack
    iv. Distinguishing attack
    v. Guess-and-Determine attack

e. Implementation and performance reports:
    i. Targeted software and/or
    ii. Targeted hardware
f. Justification on design principles of the algorithm. See **Annex C** for an example.
g. Test vectors
    - Number of keys: - a minimum of 3 for each key size
    - Number of Initialization Vectors: - 3 for each key
    - Length of keystream: - 256 bits
    - Internal state after generating 256 keystream bits

3.3 Asymmetric Cryptographic Primitive

The submission criteria are as follows:

a. Scheme:
    i. Encryption
    ii. Key agreement
    iii. Digital signature
b. Proof of correctness
c. Security analysis shall include but not limited to:
    i. Hard mathematical problems and assumptions
    ii. Minimum key length needed to achieve the security level[1] of 2^{128}
    iii. Security model and its proof[2]
d. Implementation and performance reports:
    i. Targeted software, and/or
    ii. Targeted hardware
e. Justification on design principles of the algorithm.
f. Test vectors
    - Number of key pairs: - a minimum of 3 key pairs
    - Number of processing samples for each key pairs: - 2 samples

[1] Security level means the number of steps in the best known attack on a cryptographic primitive
[2] Accepted techniques include reduction technique, game-hopping or universal composability

3.4 Cryptographic Hash Function Primitives

Cryptographic hash function primitives are divided into two categories; general-purpose cryptographic hash function and lightweight cryptographic hash function. The submission criteria are as follows:

1) Cryptographic Hash Function

a. Digest size of 224 bits, 256 bits, 384 bits, 512 bits or larger

b. Maximum message length 2^{64}-1 bits

c. Security analysis report shall include but not limited to:

   i. Pre-image resistance

   ii. Second pre-image resistance

   iii. Collision resistance

d. Implementation and performance reports:

   i. Targeted software, and/or

   ii. Targeted hardware

e. Justification on design principles of the algorithm. See **Annex D** for an example.

f. Test vectors

   - Number of samples for each data size: - 3 samples
   - Intermediate state for each round

2) Lightweight Cryptographic Hash Function

a. Digest size of 80 bits, 128 bits and 160 bits.

b. Maximum message length 2^{64}-1 bits.

c. Security analysis report shall include but not limited to:

   i. Pre-image resistance

   ii. Second pre-image resistance

   iii. Collision resistance

d. Implementation and performance reports:

   i. Targeted software and/or

      a) Program code size

      b) RAM size

   ii. Targeted hardware

      a) Chip area

b) Cycle

c) Bits per cycle

d) Power

e) Energy

e. Justification on design principles of the algorithm. See **Annex E** for an example.

f. Test vectors

- Number of samples for each data size: - 3 samples
- Intermediate state for each round

3.5 Cryptographic Key Generation Primitive

The submission criteria are as follows:

a. Scope:

i. Prime Number Generator

b. Analysis report shall include but not limited to:

i. Probabilistic Prime Generators

a) Prove at least 75% correctness (on par Miller Rabin Primality test)

b) Primality test should give an output “input is prime”, “input is composite” or “test inconclusive”

c) Run in polynomial time

d) Test vectors

- Sizes of prime: - 512, 1024, 2048, 3072, 4096, 7680, 15360 bits
- Number of seeds for each prime size: - 3 seeds (128, 256, 512 bit)

ii. Deterministic Prime Generators

a) Proof of correctness

b) Run in polynomial time

iii. Able to distinguish Carmichael numbers from prime numbers

iv. Able to generate pseudo primes samples from the generator

v. NIST statistical tests

c. Implementation and performance reports:

i. Targeted software and/or

ii. Targeted hardware

3.6 Cryptographic Pseudo Random Number Generator Primitive

The submission criteria are as follows:

a. Scope:
   i. Pseudo Random Number Generator (PRNG)

b. Analysis report shall include but not limited to:
   i. PRNG based on asymmetric methodologies
      a) Prove of correctness
      b) Run in polynomial time
      c) If a PRNG's internal state contains n bits, its period should be at least 2^n.
      d) Test vectors
         - Sizes of seed: - 128, 256, 512 bit
      e) Utilizing strong asymmetric parameters
         - Integer Factorization Problem (IFP): - 1024,2048, 4096 bit
         - Discrete Logarithm Problem (DLP): - 1024, 2048, 4096 bit
   ii. PRNG based on symmetric methodologies
      a) Run in polynomial time
      b) Test vectors
         - Sizes of seed: - 128, 256, 512 bit
      c) Utilizing strong symmetric constructions
         - AES
         - TDES
   iii. PRNG not based on asymmetric or symmetric methodologies
      a) Justification on design principles of the algorithm
      b) Proof of correctness
      c) Run in polynomial time
      d) Test vectors
         - Sizes of seed: - 128, 256, 512 bit
   iv. NIST statistical tests

c. Implementation and performance reports:
   i. Targeted software and/or
   ii. Targeted hardware

4.0 Evaluation Criteria for Each Primitive

This section describes the evaluation criteria for each primitive.

4.1 Block Cipher Primitives

The evaluation criteria are as follows:

a. Security
   - i. Achieve a security level which is comparable to its key size based on cryptanalysis attacks. For example, 128-bit security level for 128-bit key.
   - ii. Pass all NIST statistical tests based on nine data categories. See **Annex F**.

b. Cost and Performance
   - i. Block Cipher: Computational efficiency and memory requirements are comparable to AES.
   - ii. Lightweight Block Cipher: Hardware and/or software implementation measurements comparable to PRESENT.

c. Implementation Characteristics
   - i. Flexibility of algorithm may include but not limited to:
     - a) Algorithm can accommodate additional key and block sizes.
     - b) Algorithm can be implemented securely and efficiently in a wide variety of platforms and applications.
     - c) Algorithm can be operated as stream cipher, message authentication code (MAC) generator, pseudorandom number generator (PRNG) or hash function.
   - ii. Suitability of software and hardware

     It is an added advantage for the algorithm to be efficiently implemented in both software and hardware.
   - iii. Design simplicity

     A design that looks elegant, concise, neat and easy to understand would be an advantage.

d. Soundness of justification on algorithm’s design principles.

4.2 Stream Cipher Primitive

The evaluation criteria are as follows:

a. Security

   i. Achieve a security level which is comparable to its key size based on cryptanalysis attacks. For example, 128-bit security level for 128-bit key.

   ii. Pass all NIST statistical tests.

b. Cost and Performance

   Computational efficiency and memory requirements are comparable to ChaCha20.

c. Implementation Characteristics

   i. Flexibility of algorithm may include but not limited to:

      a) Algorithm can accommodate additional key size.

      b) Algorithm can be implemented securely and efficiently in a wide variety of platforms and applications.

      c) Algorithm can be operated as pseudorandom number generator (PRNG).

   ii. Suitability of software and hardware

      It is an added advantage for the algorithm to be efficiently implemented in both software and hardware.

   iii. Design simplicity

      A design that looks elegant, concise, neat and easy to understand would be an advantage.

d. Soundness of justification on algorithm's design principles.

4.3 Asymmetric Cryptographic Primitive

The evaluation criteria are as follows:

a. Security

   i. Hard mathematical problems and assumptions include but not limited to:

      a) Conventional:

         1. Discrete Logarithm Problem and variations

         2. Integer Factorization Problem and variations

      b) Post quantum:

         1. Lattice based Problems

         2. Code based Problems

        ii. Security level[3] to be at least 2^{128}
        iii. Correctness of security model and its proof[4]
    b. Cost and Performance
        i. Parameter size for each security level:
            a) Encryption – key size, ciphertext size
            b) Key agreement – key size, number of passes, bandwidth
            c) Digital signature – key size, signature size
        ii. Computational complexity
    c. Soundness of justification on algorithm's design principles.
    d. Valid proof of correctness.
    e. Comparison analysis is an advantage.

4.4 Cryptographic Hash Function Primitives

The evaluation criteria are as follows:

a. Security

   Achieve a security level which is comparable to its digest size based on cryptanalysis attacks. For example, 112-bit security level for 224-bit digest size.

b. Cost and Performance
   - i. Cryptographic Hash Function:

     Computational efficiency and memory requirements are comparable to SHA-3.
   - ii. Lightweight Cryptographic Hash Function:

     Hardware and/or software implementation measurements comparable to SPONGENT.

c. Implementation Characteristics
   - i. Flexibility of algorithm may include but not limited to:
     - a) Algorithm can accommodate additional digest size.
     - b) Algorithm can be implemented securely and efficiently in a wide variety of platforms and applications.
   - ii. Design simplicity

     A design that looks elegant, concise, neat and easy to understand would be an advantage.

d. Soundness of justification on algorithm's design principles.

[3] Security level means the number of steps in the best known attack on a cryptographic algorithm
[4] Accepted techniques include reduction technique, game-hopping or universal composability

4.5 Cryptographic Key Generation Primitive

The evaluation criteria are as follows:

a. Security
   i. Safe pseudo random prime number generator
   ii. Integer Factorization Problem (IFP) safe
      a) Strong prime number
      b) Strong prime number pair
   iii. Discrete Logarithm Problem (DLP) safe
      a) Strong prime number
   iv. Pass all NIST statistical tests.
b. Cost and Performance
   i. Run in polynomial time
c. Implementation Characteristics
   i. Flexibility of algorithm may include but not limited to:
      a) Algorithm can accommodate additional extendable block size.
      b) Generating prime within the given interval.
      c) Algorithm can be implemented efficiently in a wide variety of platforms and applications.
   ii. Suitability of software and hardware
      It is an added advantage for the algorithm to be efficiently implemented in both software and hardware.
   iii. Design simplicity
      A design that looks elegant, concise, neat and easy to understand would be an advantage.
d. Soundness of justification on algorithm's design principles.
e. Comparison analysis with existing prime number generators is an advantage.

4.6 Cryptographic Pseudo Random Number Generator Primitive

The evaluation criteria are as follows:

a. Security
   i. Safe pseudo random number generator
   ii. Asymmetric based safe
      a) Integer Factorization Problem (IFP)
      b) Discrete Logarithm Problem (DLP) safe

iii. Symmetric based safe

iv. Security analysis with respect to seed entropy.

v. Pass all NIST statistical tests.

b. Cost and Performance

  i. Run in polynomial time

c. Implementation Characteristics

  i. Flexibility of algorithm may include but not limited to:

    a) Algorithm can accommodate additional extendable block size.

    b) Algorithm can be implemented efficiently in a wide variety of platforms and applications.

  ii. Suitability of software and hardware

  It is an added advantage for the algorithm to be efficiently implemented in both software and hardware.

  iii. Design simplicity

  A design that looks elegant, concise, neat and easy to understand would be an advantage.

d. Soundness of justification on algorithm's design principles.

e. Comparison analysis with existing random number generators is an advantage.

5.0 Licensing Requirements

This section discusses the licensing requirements for submitted algorithms.

1. Submitted algorithms should, if selected by MySEAL, be available royalty-free. If this is not possible, then access is non-discriminatory (the cryptographic algorithm should not be restricted or biased on selected users only).
2. The submitter should state the position concerning intellectual property of the submitted algorithm. The intellectual property statement is as described in Section 7.0 D. This statement should be updated when necessary.

6.0 MySEAL Project Timeline

To facilitate process implementation of this whole project, a timetable for the MySEAL project is given below.

Year	Activities
2016	Q4 : Call for cryptographic algorithms
2017	Q4 : Cryptographic algorithms submission due for symmetric, asymmetric, cryptographic hash function and cryptographic key generation primitives.
2018	Q1 – Q2 : First phase of evaluation Q2 : (a) Cryptographic algorithms submission due for cryptographic pseudo random number generator primitive. (b) Announcement of the first phase shortlisted cryptographic algorithms Q3 – Q4 : Second phase of evaluation
2019	Q1 : (a) Second phase of evaluation (continuation from 2018) (b) Announcement of the second phase shortlisted cryptographic algorithms Q2 – Q4 : Final phase of evaluation Q4 : Final announcement of MySEAL
2020	Publication and promotion of MySEAL

7.0 Formal Submission Requirement

This section discusses the formal submission requirement for submitted algorithms.

1. MySEAL project will treat all submitted cryptographic algorithms confidentially as well as any related information, data and documents received for the intended purpose as spelled out in this document.
2. This project ensures the selection and handling process is carried in a confidential manner.
3. Any external experts appointed for the intended purposes are bound by an obligation of confidentiality.
4. This project reserves the right to reject submitted cryptographic algorithms that are not clearly specified and easily comprehensible or that fail to meet MySEAL requirements in some way.
5. Once the cryptographic algorithm has been submitted, submitter will not hear anything until the submitted algorithm has been selected, unless: -
   a. MySEAL project need more information or supporting documents
   b. submitter have made an enquiry, complaint or to clarify matters which required additional information

7.1 Algorithm Submission Package

The following are to be provided with any cryptographic algorithm submission:

A. Cover sheet with the following information:

[Refer to **Annex G**: Algorithm Submission Form]

1. Submission information either individual or organisation
2. Principal submitter's name, office telephone number, mobile number, fax number, e-mail address and postal address
3. Name(s) of auxiliary submitter(s) (if any)
4. Name of algorithm's inventor(s)/developer(s)
5. Name of algorithm's owner (if different from the submitter)
6. Signature of submitter
7. Organisation information (for organisation submission only)
8. Algorithm's name
9. Type of submitted algorithm, proposed security level and proposed environment.

B. Algorithm specification and supporting documentation:

1. A complete and unambiguous description of the algorithm in the most suitable form, such as a mathematical description, a textual description with diagrams or pseudo-code. The specification of an algorithm using code is not permitted. Test vectors should be in the form of hexadecimal. For asymmetric algorithms, a method for key generation and parameter selection needs to be specified.
2. A statement that there are no hidden weaknesses inserted by the designers. See **Annex G(D.2)**.
3. A statement of the claim on security properties and expected security level, together with an analysis of the algorithm with respect to standard cryptanalytic attacks. Weak keys should also be considered.
4. A statement giving the strengths and limitations of the algorithm.
5. A design rationale explaining design choices.
6. A statement of the estimated computational efficiency in software. Estimates are required for different sub-operations like key setup, primitive setup, and encryption/decryption (as far as applicable). The efficiency should be estimated both in cycles per byte and cycles per block, indicating the processor type and memory. If performance varies with the size of the inputs, then values for some typical sizes should be provided. Optionally the designers may provide estimates for performance in hardware (area, speed, gate count, a description in Hardware Description Language - HDL).
7. A description of the basic techniques for implementers to avoid implementation weaknesses.

C. Implementations and test values:

1. A sufficient number of test vectors for each parameter.
2. Reference implementation in C programming language. MySEAL will specify a set of cryptographic API applicable only for symmetric algorithms and hash function, which will be available at http://myseal.cybersecurity.my. All submissions shall implement the API so that the test system can be compatible with all the submissions.
3. Optionally, an optimized implementation for some architecture, a JAVA implementation or an assembly language implementation.

D. Intellectual Property statement:

A statement that gives the position concerning Intellectual Property and the royalty policy for the algorithm (if selected). This statement should include an undertaking to update the MySEAL project when necessary. See **Annex G(D.3)**.

7.2 Instructions for submission

1. All submissions must be in either Bahasa Melayu or English.
2. Items A, B and D shall be supplied in both paper and electronic forms. The electronic form shall be in read-only format.
3. Item C shall be supplied in electronic form in read-only format.
4. Submission shall be done in two separate discs. First disc contains item A and D. Second disc (and any subsequent discs) contains item B and C.
5. Every disc shall be labelled with the submitter's name, the name of the algorithm and the date of submission. Every disc shall contain a text file labelled "README" listing all files included on the disc with a brief description of the content of each file. Both paper submissions and optical discs should be in one sealed package and labelled as described in **Annex I**.
6. The submissions should arrive on or before 17th November 2017 at the following address:

   Sekretariat Kumpulan Fokus MySEAL
   CyberSecurity Malaysia,
   Level 7, Tower 1,
   Menara Cyber Axis,
   Jalan Impact,
   63000 Cyberjaya,
   Selangor Darul Ehsan,
   Malaysia.

   An acknowledgement will be sent by email within 3 working days upon receipt.
7. Any general questions can be forwarded to myseal.fg@cybersecurity.my. Answers to relevant questions will be posted at http://myseal.cybersecurity.my.

8.0 General Information

General information regarding MySEAL project are available at http://myseal.cybersecurity.my.

ANNEX A

Justification on Design Principles of Rijndael

The following statements were extracted from the paper titled **AES Proposal : Rijndael** to give an example statement of design principles. The full paper can be retrieved at http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf.

<u>Design rationale</u>

The three criteria taken into account in the design of Rijndael are the following:

a. Resistance against all known attacks;

b. Speed and code compactness on a wide range of platforms;

c. Design simplicity.

In most ciphers, the round transformation has the Feistel Structure. In this structure typically part of the bits of the intermediate State are simply transposed unchanged to another position. The round transformation of Rijndael does not have the Feistel structure. Instead, the round transformation is composed of three distinct invertible uniform transformations, called layers. By “uniform”, we mean that every bit of the State is treated in a similar way.

The specific choices for the different layers are for a large part based on the application of the Wide Trail Strategy, a design method to provide resistance against linear and differential cryptanalysis. In the Wide Trail Strategy, every layer has its own function:

The linear mixing layer: guarantees high diffusion over multiple rounds.

The non-linear layer: parallel application of S-boxes that have optimum worst-case nonlinearity properties.

The key addition layer: A simple EXOR of the Round Key to the intermediate State.

Before the first round, a key addition layer is applied. The motivation for this initial key addition is the following. Any layer after the last key addition in the cipher (or before the first in the context of known-plaintext attacks) can be simply peeled off without knowledge of the key and therefore does not contribute to the security of the cipher. (e.g., the initial and final permutation in the DES). Initial or terminal key addition is applied in several designs, e.g., IDEA, SAFER and Blowfish.

In order to make the cipher and its inverse more similar in structure, the linear mixing layer of the last round is different from the mixing layer in the other rounds. It can be shown that this does not improve or reduce the security of the cipher in any way. This is similar to the absence of the swap operation in the last round of the DES.

Motivation for design choices

In the following subsections, we will motivate the choice of the specific transformations and constants. We believe that the cipher structure does not offer enough degrees of freedom to hide a trap door.

The reduction polynomial $\boldsymbol{m}(\boldsymbol{x})$

The polynomial $m(x)$ ('$11B$') for the multiplication in $GF(2^8)$ is the first one of the list of irreducible polynomials of degree 8.

The ByteSub S-box

The design criteria for the S-box are inspired by differential and linear cryptanalysis on the one hand and attacks using algebraic manipulations, such as interpolation attacks, on the other:

1. Invertibility
2. Minimisation of the largest non-trivial correlation between linear combinations of input bits and linear combination of output bits
3. Minimisation of the largest non-trivial value in the EXOR table
4. Complexity of its algebraic expression in $GF(2^8)$
5. Simplicity of description

For invertible S-boxes operating on bytes, the maximum input/output correlation can be made as low as 2^{-3} and the maximum value in the EXOR table can be as low as 4 (corresponding to a difference propagation probability of 2^{-6}).

By definition, the selected mapping has a very simple algebraic expression. This enables algebraic manipulations that can be used to mount attacks such as interpolation attacks. Therefore, the mapping is modified by composing it with an additional invertible affine transformation. This affine transformation does not affect the properties with respect tot the first three criteria, but if properly chosen, allows the S-box to satisfy the fourth criterion.

We have chosen an affine mapping that has a very simple description per se, but a complicated algebraic expression if combined with the 'inverse' mapping. It can be seen as modular polynomial multiplication followed by an addition:

$$b(x) = (x^7 + x^6 + x^2 + x) + a(x)(x^7 + x^6 + x^5 + x^4 + 1) \bmod x^8 + 1$$

The modulus has been chosen as the simplest modulus possible. The multiplication polynomial has been chosen from the set of polynomials coprime to the modulus as the one with the simplest description. The constant has been chosen in such a way that that the S-box has no fixed points ($S - box(a) = a$) and no 'opposite fixed points' ($S - box(a) = \bar{a}$).

Note: other S-boxes can be found that satisfy the criteria above. In the case of suspicion of a trapdoor being built into the cipher, the current S-box might be replaced by another one. The cipher structure and number of rounds as defined even allow the use of an S-box that does not optimise the differential and linear cryptanalysis properties (criteria 2 and 3). Even an S-box that is "average" in this respect is likely to provide enough resistance against differential and linear cryptanalysis.

The MixColumn transformation

MixColumn has been chosen from the space of 4-byte to 4-byte linear transformations according to the following criteria:

1. Invertibility;
2. Linearity in $GF(2)$
3. Relevant diffusion power
4. Speed on 8-bit processors
5. Symmetry
6. Simplicity of description

Criteria 2, 5 and 6 have lead us to the choice to polynomial multiplication modulo $x^4 + 1$. Criteria 1, 3 and 4 impose conditions on the coefficients. Criterion 4 imposes that the coefficients have small values, in order of preference ‘00’, ’01’, ’02’, ’03’…The value ‘00’ implies no processing at all, for ‘01’ no multiplication needs to be executed, ‘02’ can be implemented using *xtime* and ‘03’ can be implemented using *xtime* and an additional EXOR.

The criterion 3 induces a more complicated conditions on the coefficients.

Branch number

In our design strategy, the following property of the linear transformation of MixColumn is essential. Let F be a linear transformation acting on byte vectors and let the byte weight of a vector be the number of nonzero bytes (not to be confused with the usual significance of Hamming weight, the number of nonzero bits). The byte weight of a vector is denoted by $W(a)$. The Branch Number of a linear transformation is a measure of its diffusion power:

Definition: The branch number of a linear transformation F is

$$min_{a\neq 0}(W(a) + W(F(a))).$$

A non-zero byte is called an active byte. For MixColumn it can be seen that if a state is applied with a single active byte, the output can have at most 4 active bytes, as MixColumn acts on the columns independently. Hence, the upper bound for the branch number is 5. The coefficients have been chosen in such a way that the upper bound is reached. If the branch number is 5, a difference in 1 input (or output) byte propagates to all 4 output (or input) bytes, a 2-byte input (or output) difference to at least 3 output (or input) bytes. Moreover, a linear relation between input and output bits involves bits from at least 5 different bytes from input and output.

The ShiftRow offsets

The choice from all possible combinations has been made based on the following criteria:

1. The four offsets are different and C0 = 0
2. Resistance against attacks using truncated differentials
3. Resistance against the Square attack
4. Simplicity

For certain combinations, attacks using truncated differentials can tackle more rounds (typically only one) than for other combinations. For certain combinations the Square attack can tackle more rounds than others. From the combinations that are best with respect to criteria 2 and 3, the simplest ones have been chosen.

The key expansion

The key expansion specifies the derivation of the Round Keys in terms of the Cipher Key. Its function is to provide resistance against the following types of attack:

- Attacks in which part of the Cipher Key is known to the cryptanalyst
- Attacks where the Cipher Key is known or can be chosen, e.g., if the cipher is used as the compression function of a hash function
- Related-key attacks. A necessary condition for resistance against related-key attacks is that there should not be two different Cipher Keys that have a large set of Round Keys in common.

The key expansion also plays an important role in the elimination of symmetry:

- Symmetry in the round transformation: the round transformation treats all bytes of a state in very much the same way. This symmetry can be removed by having round constants in the key schedule;
- Symmetry between the rounds: the round transformation is the same for all rounds. This equality can be removed by having round-dependent round constants in the key schedule.

The key expansion has been chosen according to the following criteria:

- It shall use an invertible transformation, i.e., knowledge of any N_k consecutive words of the Expanded Key shall allow to regenerate the whole table;
- Speed on a wide range of processors;
- Usage of round constants to eliminate symmetries;
- Diffusion of Cipher Key differences into the Round Keys;
- Knowledge of a part of the Cipher Key or Round Key bits shall not allow to calculate many other Round Key bits.
- Enough non-linearity to prohibit the full determination of Round Key differences from Cipher Key differences only;
- Simplicity of description.

In order to be efficient on 8-bit processors, a light-weight, byte oriented expansion scheme has been adopted. The application of SubByte ensures the non-linearity of the scheme, without adding much space requirements on an 8-bit processor.

Number of rounds

We have determined the number of rounds by looking at the maximum number of rounds for which shortcut attacks have been found and added a considerable security margin. (A shortcut attack is an attack more efficient than exhaustive key search.)

For Rijndael with a block length and key length of 128 bits, no shortcut attacks have been found for reduced versions with more than 6 rounds. We added 4 rounds as a security margin. This is a conservative approach, because:

- Two rounds of Rijndael provide "full diffusion" in the following sense: every state bit depends on all state bits two rounds ago, or, a change in one state bit is likely to affect half of the state bits after two rounds. Adding 4 rounds can be seen as adding a "full diffusion" step at the beginning and at the end of the cipher. The high diffusion of a Rijndael round is thanks to its uniform structure that operates on all state bits. For so-called Feistel ciphers, a round only operates on half of the state bits and full diffusion can at best be obtained after 3 rounds and in practice it typically takes 4 rounds or more.
- Generally, linear cryptanalysis, differential cryptanalysis and truncated differential attacks exploit a propagation trail through n rounds in order to attack $n + 1$ or $n + 2$ rounds. This is also the case for the Square attack that uses a 4-round propagation structure to attack 6 rounds. In this respect, adding 4 rounds actually doubles the number of rounds through which a propagation trail has to be found.

For Rijndael versions with a longer Key, the number of rounds is raised by one for every additional 32 bits in the Cipher Key, for the following reasons:

- One of the main objectives is the absence of shortcut attacks, i.e., attacks that are more efficient than exhaustive key search. As with the key length the workload of exhaustive key search grows, shortcut attacks can afford to be less efficient for longer keys.
- Known-key (partially) and related-key attacks exploit the knowledge of cipher key bits or ability to apply different cipher keys. If the cipher key grows, the range of possibilities available to the cryptanalyst increases.

As no threatening known-key or related-key attacks have been found for Rijndael, even for 6 rounds, this is a conservative margin.

For Rijndael versions with a higher block length, the number of rounds is raised by one for every additional 32 bits in the block length, for the following reasons:

- For a block length above 128 bits, it takes 3 rounds to realise full diffusion, i.e., the diffusion power of a round, relative to the block length, diminishes with the block length.
- The larger block length causes the range of possible patterns that can be applied at the input/output of a sequence of rounds to increase. This added flexibility may allow to extend attacks by one or more rounds.

We have found that extensions of attacks by a single round are even hard to realise for the maximum block length of 256 bits. Therefore, this is a conservative margin.

ANNEX B

Justification on Design Principles of PRESENT

The following statements were extracted from the paper titled **PRESENT: An Ultra-Lightweight Block Cipher** to give an example statement of design principles. The full paper can be retrieved at http:// www.ist-ubisecsens.org/publications/present_ches2007.pdf.

Design principles of PRESENT

1. Goals and environment of use
    a. The cipher is to be implemented in hardware.
    b. Applications will only require moderate security levels. Consequently, 80-bit security will be adequate. Note that this is also the position taken for hardware profile stream ciphers submitted to eSTREAM.
    c. Applications are unlikely to require the encryption of large amounts of data. Implementations might therefore be optimised for performance or for space without too much practical impact.
    d. In some applications it is possible that the key will be fixed at the time of device manufacture. In such cases there would be no need to re-key a device (which would incidentally rule out a range of key manipulation attacks).
    e. After security, the physical space required for an implementation will be the primary consideration. This is closely followed by peak and average power consumption, with the timing requirements being a third important metric.
    f. In applications that demand the most efficient use of space, the block cipher will often only be implemented as encryption-only. In this way it can be used within challenge-response authentication protocols and, with some careful state management, it could be used for both encryption and decryption of communications to and from the device by using the counter mode.

2. The permutation layer

When choosing the mixing layer, our focus on hardware efficiency demands a linear layer that can be implemented with a minimum number of processing elements, i.e. transistors. This leads us directly to bit permutations. Given our focus on simplicity, we have chosen a regular bit-permutation and this helps to make a clear security analysis.

3. The S-box

We use a single 4-bit to 4-bit S-box S: F42 → F42 in present. This is a direct consequence of our pursuit of hardware efficiency, with the implementation of such an S-box typically being much more compact than that of an 8-bit S-box. Since we use a bit permutation for the linear diffusion layer, AES-like diffusion techniques are not an option for present. Therefore, we place some additional conditions on the S-boxes to improve the so-called avalanche of change.

ANNEX C

Justification on Design Principles of CHACHA20

The following statements were extracted from the paper titled **ChaCha, a variant of Salsa20** to give an example statement of design principles. The full paper can be retrieved at https://cr.yp.to/chacha/chacha-20080120.pdf.

<u>**Design principles of ChaCha20**</u>

1. Introduction

ChaCha follows the same basic design principles as Salsa20, but I changed some of the details, most importantly to increase the amount of diffusion per round. I speculate that the minimum number of secure rounds for ChaCha is smaller than the minimum number of secure rounds for Salsa20.

This extra diffusion does not come at the expense of extra operations. A ChaCha round has 16 additions and 16 xors and 16 constant-distance rotations of 32-bit words, just like a Salsa20 round. Furthermore, ChaCha has the same levels of parallelism and vectorizability as Salsa20, and saves one of the 17 registers used by a "natural" Salsa20 implementation. So it is reasonable to guess that a ChaCha round can achieve the same software speed as a Salsa20 round—and even better speed than a Salsa20 round on some platforms. Consequently, if ChaCha has the same minimum number of secure rounds as Salsa20, then ChaCha will provide better overall speed than Salsa20 for the same level of security.

Of course, performance should be measured, not guessed! I wrote and posted new public-domain software for ChaCha, and timed that software, along with the fastest available Salsa20 software, on several computers, using the latest version (20080120) of the eSTREAM benchmarking framework.

2. The quarter-round

ChaCha, like Salsa20, uses 4 additions and 4 xors and 4 rotations to invertibly update 4 32-bit state words. However, ChaCha applies the operations in a different order, and in particular updates each word twice rather than once. Specifically, ChaCha updates a, b, c, d as follows:

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

3. The matrix

ChaCha, like Salsa20/*r*, builds a 4 × 4 matrix, invertibly transforms the matrix through *r* rounds, and adds the result to the original matrix to obtain a 16-word (64-byte) output block. There are three differences in the details. First, ChaCha permutes the order of words in the output block to match the permutation described above. This has no effect on security; it saves time on SIMD platforms; it makes no difference in speed on other platforms. Second, ChaCha builds the initial matrix with all attacker-controlled input words at the bottom.

ANNEX D

Justification on Design Principles of Keccak

The following statements were extracted from the paper titled **Keccak sponge function family main document** to give an example statement of design principles. The full paper can be retrieved at https:// keccak.noekeon.org/Keccak-main-2.1.pdf.

1) Choosing the sponge construction

Defining a generic attack:

Definition 1: A shortcut attack on a sponge function is a generic attack if it does not exploit specific properties of the underlying permutation (or transformation).

The Keccak hash function makes use of the sponge construction. This results in the following property:

Provability: It has a proven upper bound for the success probability, and hence also a lower bound for the expected workload, of generic attacks.

The design philosophy underlying Keccak is the hermetic sponge strategy. This consists of using the sponge construction for having provable security against all generic attacks and calling a permutation (or transformation) that should not have structural properties with the exception of a compact description. Additionally, the sponge construction has the following advantages over constructions that make use of a compression function:

a. **Simplicity:** Compared to the other constructions for which upper bounds have been proven for the success of generic attacks, the sponge construction is very simple, and it also provides a bound that can be expressed in a simple way.
b. **Variable-length output:** It can generate outputs of any length and hence a single function can be used for different output lengths.
c. **Flexibility:** Security level can be incremented at the cost of speed by trading in bitrate for capacity, using the same permutation (or transformation).
d. **Functionality:** Thanks to its long outputs and proven security bounds with respect to generic attacks, a sponge function can be used in a straightforward way as a MAC function, stream cipher, a re-seedable pseudorandom bit generator and a mask generating function.

To support arbitrary bit strings as input, the sponge construction requires a padding function. We refer to Section 3.2 of Keccak sponge function family main document for a rationale for the specific padding function we have used.

2) Choosing an iterated permutation

The sponge construction requires an underlying function f, either a transformation or a permutation. f should be such that it does not have properties that can be exploited in shortcut attacks. We have chosen a permutation, constructed as a sequence of almost identical rounds because of the following advantages:

a. **Block cipher experience:** An iterated permutation is an iterated block cipher with a fixed key. In its design one can build on knowledge obtained from block cipher design and cryptanalysis.
b. **Memory efficiency:** Often a transformation is built by taking a permutation and adding a feedforward loop. This implies that (at least part of) the input must be kept during the complete computation. This is not the case for a permutation, leading to a relatively small RAM footprint.
c. **Compactness:** Iteration of a single round leads to a compact specification and potentially compact code and hardware circuits.

3) Designing the Keccak-f permutations

The design criterion for the Keccak-f permutations is to have no properties that can be exploited in a shortcut attack when being used in the sponge construction. It is constructed as an iterated block cipher similar to Noekeon and Rijndael, with the key schedule replaced by some simple round constants. Here we give a rationale for its features:

a. **Bit-oriented structure Attacks:** Where the bits are grouped (e.g., in bytes), such as integral cryptanalysis and truncated trails or differentials, are unsuitable against the Keccak-f structure.
b. **Bitwise logical operations and fixed rotations:** Dependence on CPU word length is only due to rotations, leading to an efficient use of CPU resources on a wide range of processors. Implementation requires no large tables, removing the risk of table-lookup based cache miss attacks. They can be programmed as a fixed sequence of instructions, providing protection against timing attacks.
c. **Symmetry**: This allows to have very compact code in software and a very compact co-processor suitable for constrained environments.
d. **Parallelism**: Thanks to its symmetry and the chosen operations, the design is well-suited for ultra-fast hardware implementations and the exploitation of SIMD instructions and pipelining in CPUs.
e. **Round degree 2:** This makes the analysis with respect to differential and linear cryptanalysis easier, leads to relatively simple (albeit large) systems of algebraic equations and allows the usage of very powerful protection measures against differential power analysis (DPA) both in software and hardware that are not suited for most other nonlinear functions.
f. **Matryoshka structure:** The analysis of small versions is relevant for larger versions.
g. **Eggs in another basket:** The choice of operations is very different from that in SHA-1 and the members of the SHA-2 family on the one hand and from AES on the other.

4) Choosing the parameter values

In Keccak, there are basically three security-relevant parameters that can be varied:

a. b: width of Keccak-f,
b. c: capacity, limited by $c < b$,
c. n_r: number of rounds in Keccak-f.

The parameters of the candidate sponge functions have been chosen for the following reasons.

a. $c = 2n$: for the fixed-output-length candidates, we chose a capacity equal to twice the output length n. This is the smallest capacity value such that there are no generic attacks with expected complexity below 2^n.
b. $b = 1600$: The width of the Keccak-f permutation is chosen to favor 64-bit architectures while supporting all required capacity values using the same permutation.
c. Parameters for Keccak[]: for the variable-output-length candidate Keccak[], we chose a rate value that is a power of two and a capacity not smaller than 512 bits and such that their sum equals 1600. This results in $r = 1024$ and $c = 576$. This capacity value precludes generic attacks with expected complexity below 2288. A rate value that is a power of two may be convenient in some applications to have a block size which is a power of two, e.g., for a real-time application to align its data source (assumed to be organized in blocks of size a power of two) to the block size without the need of an extra buffer.
d. $n_r = 24$: The value of n_r has been chosen to have a good safety margin with respect to even the weakest structural distinguishers and still have good performance.

5) The difference between version 1 and version 2 of Keccak

For the 2nd round of the SHA-3 competition, we decided to modify Keccak. There are basically two modifications: the increase of the number of rounds in Keccak-f and the modification of the rate and capacity values in the four fixed-output-length candidates for SHA-3:

a. Increasing the number of rounds of Keccak-f from $12 + l$ to $12 + 2l$ (from 18 to 24 rounds for Keccak-f[1600]): this modification is due to the distinguishers that work on reduced-round variants of Keccak-f[1600] up to 16 rounds. In the logic of the hermetic sponge strategy, we want the underlying permutation to have no structural distinguishers. Sticking to 18 rounds would not contradict this strategy but would leave a security margin of only 2 rounds against a distinguisher of Keccak-f. In addition, we do think that this increase in the number of rounds increases the security margin with respect to distinguishers of the resulting sponge functions and attacks against those sponge functions.
b. For applications where the bitrate does not need to be a power of 2, the new parameters of the fixed-output-length candidates take better advantage of the performance-security trade-offs that the Keccak sponge function allows.

ANNEX E

Justification on Design Principles of SPONGENT

The following statements were extracted from the paper titled **SPONGENT: The Design of Lightweight Cryptographic Hashing** to give an example statement of design principles. The full paper can be retrieved at https://eprint.iacr.org/2011/697.pdf.

The overall design approach for SPONGENT is to target low area while favoring simplicity. The 4-bit S-box is the major block of functional logic in a serial low-area implementation of SPONGENT. It fulfills the present design criteria in terms of differential and linear properties. Moreover, any linear approximation over the S-box involving only single bits both in the input and output masks is unbiased. This aims to restrict the linear hull effect discovered in round-reduced PRESENT.

The function of the bit permutation pLayer is to provide good diffusion, by acting together with the S-box, while having a limited impact on the area requirements. This is its main design goal, while a bit permutation may occupy additional space in silicon. The counters lCounter and ɹǝʇunoƆl are mainly aimed to prevent sliding properties and make prospective cryptanalysis approaches using properties like invariant subspaces more involving.

The structures of the bit permutation and the S-box in SPONGENT make it possible to prove the following differential property:

Theorem 1: Any 5-round differential characteristic of the underlying permutation of SPONGENT with $b \geq 64$ has a minimum of 10 active S-boxes. Moreover, any 6-round differential characteristic of the underlying permutation of SPONGENT with $b \geq 256$ has a minimum of 14 active S-boxes.

The concept of counting active S-boxes is central to the differential cryptanalysis. The minimum number of active S-boxes relates to the maximum differential characteristic probability of the construction. Since in the hash setting there are no random and independent key values added between the rounds, this relation is not exact (in fact that it is even not exact for most practical keyed block ciphers). However, differentially active S-boxes are still the major technique used to evaluate the security of SPN-based hash functions.

An important property of the SPONGENT S-box is that its maximum differential probability is 2^{-2}. This fact and the assumption of the independency of difference propagation in different rounds yield an upper bound on the differential characteristic probability of 2^{-20} over 5 rounds and of 2^{-28} over 6 rounds for $b \geq 256$ which follows from the claims of Theorem 1.

Theorem 1 is used to determine the number *R* of rounds in permutation π_b: *R* is chosen in a way that π_b provides at least *b* active S-boxes.

ANNEX F

Data Categories for Block Cipher

Ciphertext produced from block ciphers only contains sequence of bits whose length is the block size of the block cipher (e.g ciphertext of LBlock Cipher is 64-bit). However, to evaluate the randomness of cryptographic algorithm, it is important to ensure that ciphertext produced contains a large sequence of bit. To achieve this, Data Categories are used to generate inputs (plaintext or key) for block ciphers to produce ciphertexts that will be concatenated in a certain way. Detail description of the nine categories of data used in block ciphers are provided below:

i. Strict Key Avalanche (SKA)
ii. Strict Plaintext Avalanche (SPA)
iii. Plaintext / Ciphertext Correlation (PCC)
iv. Cipher Block Chaining Mode (CBCM)
v. Random Plaintext / Random Key (RPRK)
vi. Low Density Key (LDK)
vii. High Density Key (HDK)
viii. Low Density Plaintext (LDP)
ix. High Density Plaintext (HDP)

1. <u>Strict Key Avalanche (SKA)</u>

The Strict Key Avalanche data category is used to examine the sensitivity of block ciphers to changes in the $x-$bit key. For a fixed plaintext block, avalanche effect is satisfied when any bit of the key is complemented, each bit of the ciphertext block changes with a probability of one half.

Each sample for this data category utilizes plaintext that is set to all zero, and X blocks of random $x-$bit base-keys. The all zero plaintext is first encrypted using each base-key. Next, each base-key is flipped at the i^{th} bit, for $1 \leq i \leq x$ giving a total of $(X * x)$ perturbed-keys. The all-zero plaintext is then encrypted using each perturbed-key. All resultant ciphertexts using pertubed-keys are XORed with the ciphertext resulting from the encryption using its corresponding base-key. Output product of the XOR operation is called a derived block which will be concatenated to construct a large sequence of bits.

2. Strict Plaintext Avalanche (SPA)

The Strict Plaintext Avalanche data category is used to examine the sensitivity of block ciphers to changes in the y −bit plaintext. For a fixed key, avalanche effect is satisfied when any bit of the plaintext is complemented, each bit of the ciphertext block changes with a probability of one half.

Each sample for this data category utilizes key that is set to all zero, and Y blocks of random y −bit base-plaintexts. Each base-plaintext is first encrypted using the all-zero key. Next, each base-plaintext is flipped at the i^{th} bit, for $1 \leq i \leq y$ giving a total of $(Y * y)$ perturbed-plaintexts. Each perturbed-plaintext is then encrypted using the all-zero key. All resultant ciphertexts of pertubed-plaintexts are XORed with the ciphertext resulting from the encryption of its corresponding base-plaintext. Output product of the XOR operation is called a derived block which will be concatenated to construct a large sequence of bits.

3. Plaintext / Ciphertext Correlation (PCC)

The Plaintext / Ciphertext Correlation data category is used to examine the correlation between plaintext / ciphertext pairs and is computed using ECB mode of operation.

Each sample for this data category utilizes Y blocks of random y −bit plaintext and one random x −bit key. Each plaintext block is encrypted using the random x −bit key. The resultant ciphertext is XORed with its corresponding plaintext. Output product of the XOR operation is called a derived block which will be concatenated to construct a large sequence of bits.

4. Cipher Block Chaining Mode (CBCM)

The Ciphertext Block Chaining Mode data category is computed using CBC mode of operation. In this data category each block of plaintext is XORed with the previous ciphertext block before being encrypted, whereas the first block is XORed with an initialization vector. A one-bit change in any plaintext or the initialization vector will affect all following ciphertext blocks.

Each sample for this data category utilizes plaintext that is set to all zero (P), a random x −bit key (K), and an all-zeroes initialization vector (IV). The encryption process is applied for I times. Derived blocks for this data category are ciphertext blocks computed in CBC mode of operation. The first ciphertext block, C_1 is define as $C_1 = E_K(IV \oplus P_1)$, whereas subsequent ciphertext blocks is define as $C_i = E_K(C_{i-1} \oplus P_i)$ for $1 \leq i \leq I$.

5. Random Plaintext / Random Key (RPRK)

The Random Plaintext / Random Key data category is used to examine the randomness of ciphertext based on random plaintext and random key. Each sample for this data category utilizes Y

blocks of random y −bit plaintext and one random x-bit key. Each plaintext block is encrypted using the random x −bit key. Derived blocks for this data category are ciphertext blocks computed in ECB mode of operation that will be concatenated to construct a large sequence of bits.

6. Low Density Key (LDK)

The Low Density Keys data category is formed based on low-density x −bit keys. Each sample for this data category utilizes Y blocks of random y-bit plaintext and X blocks of specific x −bit key. The first plaintext block is encrypted using an all-zeroes x −bit key. Then, plaintext blocks are encrypted using x-bit key with a single ‘1’ in each of the x-bit position of the key and all other key bits are set to ‘0’. This will produce Y_1 blocks of ciphertext. Next, plaintext blocks are encrypted using x-bit key with two ‘1’s in each combination of two bit positions of the key and all other key bits are set to ‘0’. This will produce C_r^n blocks of ciphertext, where $n = x$ and $r = 2$. In total, derived blocks for this data category are $Y = 1 + Y_1 + C_2^x$ ciphertext blocks computed in ECB mode of operation, and will be concatenated to construct a large sequence of bits.

7. High Density Key (HDK)

The High Density Keys data category is formed based on high-density x −bit keys. Each sample for this data category utilizes Y blocks of random y −bit plaintext and X blocks of specific x −bit key. The first plaintext block is encrypted using an all-ones x −bit key. Then, plaintext blocks are encrypted using x −bit key with a single ‘0’ in each of the x −bit position of the key and all other key bits are set to ‘1’. This will produce Y_1 blocks of ciphertext. Next, plaintext blocks are encrypted using x −bit key with two ‘0’s in each combination of two bit positions of the key and all other key bits are set to ‘1’. This will produce C_r^n blocks of ciphertext, where $n = x$ and $r = 2$. In total, derived blocks for this data category are $Y = 1 + Y_1 + C_2^x$ ciphertext blocks computed in ECB mode of operation, and will be concatenated to construct a large sequence of bits.

8. Low Density Plaintext (LDP)

The Low Density Plaintext data category is formed based on low-density y −bit plaintext blocks. Each sample for this data category utilizes X blocks of random x −bit keys and Y blocks of specific y −bit plaintext blocks. Firstly, the all-zeroes y −bit plaintext block is encrypted using the first random x −bit key. Then, plaintext blocks with a single '1' in each of the y-bit position of the plaintext and all other plaintext bits are set to '0', is encrypted using other random x −bit keys. This will produce X_1 blocks of ciphertext. Next, plaintext blocks with two ‘1’s in each combination of two bit positions of the plaintext and all other plaintext bits are set to ‘0’, is encrypted using other random x −bit keys.

This will produce C_r^n blocks of ciphertext, where $n = y$ and $r = 2$. In total, derived blocks for this data category are $X = 1 + X_1 + C_2^y$ ciphertext blocks computed in ECB mode of operation, and will be concatenated to construct a large sequence of bits.

9. <u>High Density Plaintext (HDP)</u>

The High Density Plaintext data category is formed based on high-density y −bit plaintext blocks. Each sample for this data category utilizes X blocks of random x −bit keys and Y blocks of specific y −bit plaintext blocks. Firstly, the all-zeroes y −bit plaintext block is encrypted using the first random x −bit key. Then, plaintext blocks with a single '0' in each of the y −bit position of the plaintext and all other plaintext bits are set to '1', is encrypted using other random x −bit keys. This will produce X_1 blocks of ciphertext. Next, plaintext blocks with two ‘0’s in each combination of two bit positions of the plaintext and all other plaintext bits are set to ‘1’, is encrypted using other random x −bit keys. This will produce C_r^n blocks of ciphertext, where $n = y$ and r = 2. In total, derived blocks for this data category are $X = 1 + X_1 + C_2^y$ ciphertext blocks computed in ECB mode of operation, and will be concatenated to construct a large sequence of bits.

ANNEX G

BORANG PENYERAHAN ALGORITMA

Algorithm Submission Form

SENARAI ALGORITMA KRIPTOGRAFI TERPERCAYA NEGARA (MySEAL)

MAKLUMAT SERAHAN *Submission Information*	
☐ Individu *Individual*	☐ Organisasi *Organisation*

A. MAKLUMAT PENYERAH

Submitter Information

MAKLUMAT PENYERAH *Submitter Information*	
NAMA PENYERAH UTAMA *Principal Submitter's Name*	
NO TELEFON PEJABAT *Office Tel No*	
NO TELEFON MUDAH ALIH *Mobile No*	
NO FAKSIMILI *Fax No*	
ALAMAT E-MEL *E-mail Address*	
ALAMAT SURAT MENYURAT *Postal Address*	

NAMA PENYERAH TAMBAHAN (jika berkenaan) *Name of Auxiliary Submitter(s)* *(if any)*	
NAMA PEREKA CIPTA / PEMBANGUN ALGORITMA *Name of Algorithm Inventor(s) / Developer(s)*	
NAMA PEMILIK ALGORITMA (jika berlainan daripada penyerah utama) *Name of Algorithm's Owner* *(if different from the submitter)*	
TANDATANGAN PENYERAH *Signature of Submitter*	

MAKLUMAT ORGANISASI (untuk serahan organisasi sahaja) *Organisation Information (for organisation submission only)*	
ORGANISASI *Organisation*	
ALAMAT *Address*	

B. MAKLUMAT ALGORITMA

Algorithm Information

NAMA ALGORITMA *Name of algorithm*	
PRIMITIF ALGORITMA KRIPTOGRAFI *Cryptographic Algorithm Primitive*	☐ *a) Symmetric Block Cipher* ☐ *Block Cipher* ☐ *Lightweight Block Cipher* ☐ *b) Symmetric Stream Cipher* ☐ *Synchronous stream cipher* ☐ *Self- Synchronous stream cipher* ☐ *c) Asymmetric* ☐ *Encryption* ☐ *Key agreement* ☐ *Digital signature* ☐ *d) Cryptographic Hash Function* ☐ *Cryptographic hash function* ☐ *Lightweight hash function* ☐ *e) Cryptographic Key Generation* ☐ *f) Cryptographic Pseudo Random Number Generator Primitive*
CADANGAN TAHAP KESELAMATAN *Proposed Security Level*	☐ 40 – bit ☐ 80 – bit ☐ 128 – bit ☐ 192 – bit ☐ 256 – bit ☐ Lain-lain. Sila nyatakan. *Other(s). Please specify* __________________________
CADANGAN PLATFORM *Proposed Environment*	☐ Perkakasan *Hardware* ☐ Perisian *Software*

C. MAKLUMAT TAMBAHAN

Additional Information

PENGGUNAAN *Implementation*	☐ *Bluetooth* ☐ *Global System for Mobile communications (GSM)* ☐ *Radio-Frequency Identification (RFID)* ☐ *Smart cards* ☐ *Mobile devices* ☐ *Sensor network* ☐ *Microcontroller* ☐ *Microprocessor* ☐ *Field-programmable gate array (FPGA)* ☐ Lain-lain. Sila nyatakan. *Other(s). Please specify* _________________________

D. PENYATAAN PENYERAH

Statement by the Submitter

1. Submission statement

☐ I/We do hereby understand that my/our submitted algorithm may not be selected for inclusion in MySEAL. I/We also understand and agree that after the close of the submission period, my/our submission may not be withdrawn. I/We further understand that I/we will not receive financial compensation from MySEAL project for my/our submission.

2. Statement that there are no hidden weaknesses in the algorithm design

☐ I/We certify that, to the best of my knowledge, I/we have fully disclosed there are no hidden weaknesses in my/our algorithm.

☐ I/We hereby enclosed information on the known weaknesses of my/our algorithm [……………………………………………… (file/attachment name)]

3. Intellectual Property Statement for the Submission of ……………………………………………… [*name of algorithm*] to the MySEAL Project

☐ ……………………………………………… [Submitter] currently has patents pending / has not filed for patents on the ……………………………………………… [name of algorithm]. The ……………………………………………… [name of algorithm] is provided royalty-free for commercial and non-commercial use in non-embedded applications. Licenses for use of the ……………………………………………… [name of algorithm] in embedded applications may be obtained from ……………………………………………… [name]. Aside from legal restrictions applying to encryption algorithms (if any), these licenses will be issued on a non-discriminatory basis. We will undertake to update the MySEAL project when necessary.

Diserahkan oleh (Tandatangan & Cop):
Submitted by (Signature & Stamp):
 	
Tarikh:
Date:
	Diterima oleh (Tandatangan & Cop):
	Received by (Signature & Stamp):
	
	Tarikh:
	Date:

ANNEX H

SENARAI SEMAK

Checklist

Bil *No*	**Perkara / Dokumen diperlukan** ***Document(s) needed***	**Disertakan oleh Penyerah (√)** ***Supplied by Submitter (√)***	**Disemak oleh Penerima (√)** ***Checked by Receiver (√)***	**Catatan** ***Notes***
1	Profil Syarikat *Company Profile*			
2	Laporan analisis *Analysis Report* *a) Symmetric Block Cipher* ☐ Ujian statistik NIST *NIST statistical tests* ☐ *Linear cryptanalysis* ☐ *Differential cryptanalysis* ☐ Lain-lain. Sila nyatakan. *Other(s). Please specify.* ________________________ *b) Symmetric Stream Cipher* ☐ Ujian statistik NIST *NIST statistical tests* ☐ *Algebraic attack* ☐ *Correlation attack* ☐ *Distinguishing attack* ☐ *Guess-and-Determine attack* ☐ Lain-lain. Sila nyatakan. *Other(s). Please specify.* ________________________			

Bil *No*	**Perkara / Dokumen diperlukan** ***Document(s) needed***	**Disertakan oleh Penyerah (√)** ***Supplied by Submitter (√)***	**Disemak oleh Penerima (√)** ***Checked by Receiver (√)***	**Catatan** ***Notes***
	c) Asymmetric Cryptographic Algorithm ☐ *Hard Mathematical Problems and assumptions* ☐ *Security Model and its proof* ☐ Lain-*lain*. Sila nyatakan. *Other(s). Please specify.* _______________________ *d) Cryptographic Hash Function* ☐ *Pre-image resistance* ☐ *Second pre-image resistance* ☐ *Collision resistance* ☐ Lain-lain. *Sila* nyatakan. *Other(s). Please specify.* _______________________ *e) Cryptographic Key Generation* ☐ *Probabilistic Prime Generators* ☐ *Deterministic Prime Generators* ☐ *Distinguishing Carmichael numbers from prime numbers* ☐ *Generation of pseudo primes samples from the generator* ☐ Ujian statistik NIST *NIST statistical tests* ☐ Lain-lain. Sila nyatakan. *Other(s). Please specify.* _______________________			

Bil *No*	**Perkara / Dokumen diperlukan** ***Document(s) needed***	**Disertakan oleh Penyerah (√)** ***Supplied by Submitter (√)***	**Disemak oleh Penerima (√)** ***Checked by Receiver (√)***	**Catatan** ***Notes***
	f) Cryptographic Pseudo Random Number Generator Primitive ☐ *PRNG based on asymmetric methodologies* ☐ *PRNG based on symmetric methodologies* ☐ *PRNG not based on asymmetric or symmetric methodologies* ☐ Ujian statistik NIST *NIST statistical tests* ☐ *Lain-lain. Sila nyatakan. Other(s). Please specify.* ________________________			
3	Laporan prestasi algoritma mengikut keupayaan perkakasan dan/atau perisian *Implementation and performance reports on hardware and/or software*			
4	Laporan reka bentuk *Justification on design principles*			
5	Vektor ujian *Test vectors*			
6	Penyata / perjanjian / pendedahan Harta Intelek *Intellectual Property statements / agreements / disclosures*			

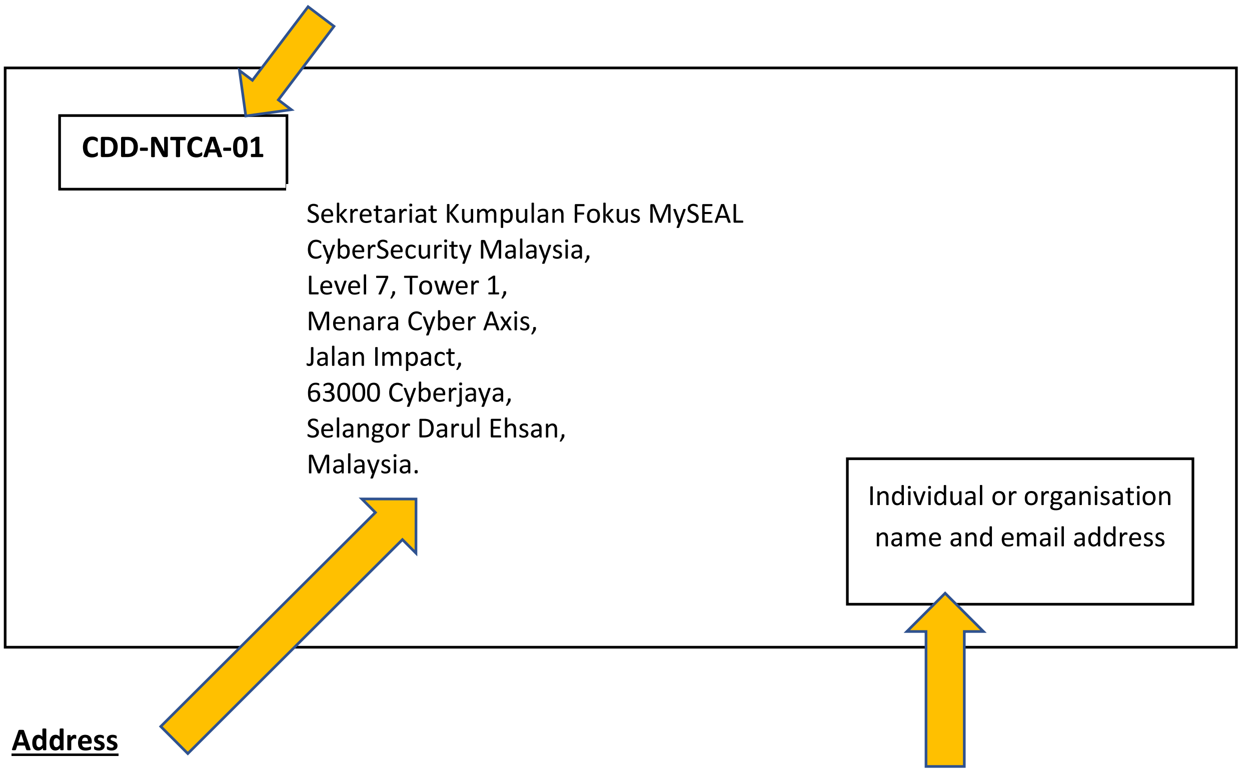
ANNEX I

LABEL GUIDELINE FOR SUBMISSION PACKAGE OF MySEAL PROJECT

PACKAGE FRONT

Project Code

Write this code on your package



CDD-NTCA-01

Sekretariat Kumpulan Fokus MySEAL
CyberSecurity Malaysia,
Level 7, Tower 1,
Menara Cyber Axis,
Jalan Impact,
63000 Cyberjaya,
Selangor Darul Ehsan,
Malaysia.

Individual or organisation name and email address

Address

Official address for submission

Submitter's Information

Write down submitter or organisation name and e-mail address