ISO/IEC JTC 1/SC 27 **N14908**

SC 27/W 2 N1021

REPLACES:

ISO/IEC JTC 1/SC 27

Information technology - Security techniques

Secretariat: DIN, Germany

DOC TYPE:	Standing Document
TITLE:	**First edition of SC 27/WG 2 Standing Document 4 -- Analysis and status of cryptographic algorithms**
SOURCE:	Project editors and co-editor (Shin'ichiro Matsuo, Matt Henricksen and Liqun Chen)
DATE:	2014-12-22
PROJECT:	**WG 2 SD4**
STATUS:	This document is being circulated within SC 27 for information. **Please note: This document is freely available from the public SC 27 public website at: http://jtc1sc27.din.de/sbe/wg2sd4**
ACTION:	FYI
DUE DATE:	
DISTRIBUTION:	P-, O- and L-Members W. Fumy, SC 27 Chairman M. De Soete, SC 27 Vice Chair E. J. Humphreys, T. Chikazawa, M. Bañón, J. Amsenga, K. Rannenberg, WG-Convenors
MEDIUM:	http://isotc.iso.org/livelink/livelink/open/jtc1sc27
NO. OF PAGES:	1 + 1 + 29

ISO/IEC JTC 1/SC 27/WG 2 N1021

REPLACES:

ISO/IEC JTC 1/SC 27/WG 2

Cryptography and security mechanisms

Convenorship: JISC (Japan)

Document type:	**Standing Documentt**
Title:	**First edition of SC 27/WG 2 Standing Document 4 -- Analysis and status of cryptographic algorithms**
Status:	**In accordance with Recommendations 01 and 21 (contained in SC 27/WG 2 N1043) of the 49th SC 27/WG 2 meeting at Mexico City, Mexico on 20th - 24th October, 2014, this document is published on the SC 27 public website at: http://jtc1sc27.din.de/sbe/wg2sd4** **This document is being circulated to experts of National Bodies and liaison organizations for information.**
Date of document:	2014-12-16
Source:	Project editors and co-editor (Shin'ichiro Matsuo, Matt Henricksen and Liqun Chen)
Expected action:	INFO
No. of pages:	1 + 29
Email of secretary:	sc27wg2-secretary@ipa.go.jp
Committee URL:	http://isotc.iso.org/livelink/livelink/open/jtc1sc27wg2

ISO/IEC JTC1 SC 27/WG 2 Standing Document 4: Analysis and Status of Cryptographic Algorithm

Shin'ichiro Matsuo, Matt Henriksen and Liqun Chen

December 16, 2014

1 Scope of this document

ISO/IEC JTC 1/SC 27/WG 2 has prepared a large number of IEC/IEC standards in the area of "Cryptography and Security Mechanisms". This standing document of the working group provides a information on security and implementation analysis of the cryptographic mechanisms specified in these standards.

This document aims to describe the information for every mechanism specified in all the standards prepared by this working group. However, this is a long-term project. In the current version of this document, the coverage is restricted to ISO/IEC 18033: "Encryption Algorithms". In the future, this document will be extended to cover other standards.

This documents contains the following three parts. Each part contains separate discussions of ISO/IEC 18033-2, ISO/IEC 18033-3 and ISO/IEC 18033-4.

Dictionary of attacks on 18033 encryption algorithms This part a contains taxonomy of attacks, and for each attack type: the name, the result of success, and the conditions under which it can succeed. This dictionary includes definitions of both academic and practical attacks. However, it does not explicitly distinguish between the two types. This document includes all discussed attacks in academic community such as related key attack and biclique attack.

Security status of 18033 encryption algorithms This part contains large tables which figure all state-of-the-art attacks for 18033 encryption algorithms, which include existence of attacks and required computational complexity to success the attack for adversary for each attack shown in the dictionary This part also describes explanations of security status of each algorithm, that is, short descriptions on the existing attacks and references of the attacks.

Implementation status of 18033 encryption algorithms This part contains performance results and implementation size of hardware. In performance results, the target platforms are Intel Processor (for PC), ARM Processor (for mobile device), and smartcard. In implementation size, gate counts for Low-power devices include RFID-tags and smartcard are described.

2 Dictionary of attacks on 18033 encryption algorithms

2.1 Asymmetric ciphers - 18033-2

2.1.1 Attacking Scenario

The security of asymmetric cipher is considered against "adaptive chosen cipher text attack." This attack is performed as following scenario.

Stage 1: The key generation algorithm is run, generating a public key and private key. The adversary, of course, obtains the public key, but not the private key.

Stage 2: The adversary makes a series of arbitrary queries to a decryption oracle. Each query is a label/ciphertext pair (L, C) that is decrypted by the decryption oracle, making use of the private key. The resulting decryption is given to the adversary; moreover, if the decryption algorithm fails, then this information is given to the adversary, and the attack continues. The adversary is free to construct these label/ciphertext pairs in an arbitrary way - it is certainly not required to compute them using the encryption algorithm.

Stage 3: The adversary prepares a label L^* and two "target" plaintexts M_0, M_1 of equal length, and gives these to an encryption oracle. If the scheme supports any encryption options, the adversary also chooses these. The encryption oracle chooses $b \in \{0, 1\}$ at random, encrypts M_b with label L^*, and gives the resulting "target" ciphertext C^* to the adversary.

Stage 4: The adversary continues to submit label/ciphertext pairs (L, C) to the decryption oracle, subject only to the restriction that $(L, C) \neq (L^*, C^*)$.

Stage 5: The adversary outputs $\widehat{b} \in \{0, 1\}$, and halts.

The security of the asymmetric cipher is evaluated by indistinguishability against adaptive chosen cipher text attack (IND-CCA2). The "indistinguishability" means that the adversary cannot guess the bit b chosen in the stage 3 with non-negligible probability. The game to guess b is successful in 1/2 probability. Thus, the advantage of the adversary A for asymmetric cipher AC is defined as follows.

$$Adv_{AC}(A) = |Pr[\widehat{b} = b] - 1/2|$$

Security means that this advantage is negligible for all efficient adversaries. The security of asymmetric cipher is evaluated by assuming the underlying mathematical problem is hard. That is, mathematical rigorous proof should be given that the advantage is negligible if the underlying mathematical problem is hard.

2.1.2 Security in underlying mathematical hard problem

As wrote above, most of asymmetric ciphers are constructed based on mathematical hard problem. The security proof of asymmetric ciphers are proof of relationship between construction of the asymmetric cipher and mathematical hard problem. If the underlying mathematical problem become weak, the security of the asymmetric cipher becomes weak. Thus, we must pay attention to the underlying mathematical problems.

2.2 Block ciphers - 18033-3

The following attacks apply to some of the block ciphers listed in this document. The attacks can be considered successful when they have a complexity of less than that of a brute-force search to identify the master key used to encrypt plaintext or decrypt ciphertext. There is no consensus on how to measure the complexity of an attack. For example, a brute-force attack on a block cipher that uses a 128-bit master key has average time complexity 2^{127} and requires negligible memory and just a few ciphertexts. But many would consider an attack with average time complexity to 2^{120} to be successful even if memory requirement is 2^{100} and the attack requires the entire codebook of the block cipher.

There are many assumptions that can be made about the conditions for attacks on block ciphers to apply, including whether the attacker only knows ciphertext, some plaintext, can choose some plaintext to be encrypted under a key, or even knows the relationship between keys (without knowing the value of the keys themselves) during successive runs of the cipher.

2.2.1 Meet-in-the-middle attack

The meet-in-the-middle attack treats the iterated function of a block cipher E as separate sub-functions (e.g. $E = f_2 \cdot f_1$), and attempts to find the meeting point between one sub-function executed in the forward direction, and another in the backwards direction. It works effectively when the sub-functions rely on different bits of the master key. A good example is Triple-DES, which uses two or three keys for its three iterations of the DES sub-functions. For a meet-in-the middle attack where the key subsets for f_1 and f_2 respectively have lengths of l and m, the attack complexity converges towards $2^{(l+m)}$ rather than 2^{lm}.

The biclique attack is a sophisticated form of the meet-in-the-middle attack.

2.2.2 Related-key attack

In a related key attack, the attacker considers two (or more) instantiations of the block cipher where there is a known relation between keys, $\Delta K = K \oplus K'$. Note that the attacker is not assumed to know the value of the keys. When the key schedule is sufficiently simple, controlling the relationship between keys may lead to a relationship between bits in the intermediate texts of the encryption algorithm. Because in reality, key management protocols do not permit the attacker to precisely determine the relationship between keys in different runs, many consider this type of attack to be impractical, to the extent that many block ciphers are designed with very light key or no key schedules. Others, however, believe that the key schedule should provide protection against this type of attack. Attacks that do not use related keys are called single-key attacks.

2.2.3 Differential cryptanalysis

Differential cryptanalysis studies the effect of non-linear components on the differences in groups (usually pairs) of inputs, and the transition of those differences to the components' outputs. Given input/output differences only occur for a subset of values of non-linear components, and differential cryptanalysis helps to isolate those values. The relationship between input and output is modelled by a differential that holds with some specified probability for a pair of texts. Differential cryptanalysis is the work-horse of cryptanalysis of block ciphers, and there are many varieties.

Higher-order differential cryptanalysis is a generalization of differential cryptanalysis (which in this context can be considered as using first-order differential) that applies to more than a pair of texts. It works best on ciphers that can be represented by equations with low-order degree. The boomerang and amplified boomerang are sophisticated differential attacks that use multiple pairs of text in conjunction with differentials that extend partially rather than wholly over the cipher.

Impossible differential cryptanalysis uses differentials that cover some number of rounds, such that the output of one differential can never match the input of another. Then by collecting pairs of texts that match the input and output of the concatenated differentials, the key values that support the impossible differentials can be disqualified.

2.2.4 Linear Cryptanalysis

In linear cryptanalysis, non-linear components in the cipher are approximated by linear or affine components that hold with some probability. These approximations can be glued together with the linear components to make an approximation for nearly all of the rounds in the cipher. The attacker guesses key bits in the rounds that are not covered by the approximation, along with a bit in the approximation. Observing for how many texts the approximation holds true allows the guessed key bits to be validated.

2.3 Stream ciphers - 18033-4

The following attacks apply to some of the stream ciphers listed in this document. The attacks can be considered successful when they have a complexity less than that of a brute-force search to identify the key used to initialize the state of the stream cipher, or to identify the state of the stream cipher, or to predict future sequences of keystream. There is no consensus on how to exactly measure the complexity of an attack. For example, a brute-force attack on a stream cipher that is keyed with a 128-bit key has average time complexity 2^{127}, and requires negligible memory and just a few words of keystream. But many would consider an attack with average time complexity 2^{120} to be successful even if memory requirement is 2^{100} and the attack requires a sequence of 2^{64} bits of keystream generated under the same key-IV pair. Yet it is clear that it is more practical to launch the first attack described here.

2.3.1 Distinguishing attack

A distinguishing attack is a type of attack that is able to detect the keystream of a stream cipher is not generated by a random process. Furthermore, it might be able to use biases in the keystream to determine the nature of the stream cipher that generated the keystream. Distinguishing attacks are less useful than key recovery attacks, although in some cases they might be able to be converted with additional complexity.

2.3.2 Differential cryptanalysis

Differential cryptanalysis studies the effect of non-linear components on the differences in groups (usually pairs) of inputs, and the transition of those differences to the components' outputs. Given input/output differences only occur for a subset of values of non-linear components, and differential cryptanalysis helps to isolate those values. In the context of stream ciphers, particularly those that accept plaintext input that is able to be manipulated by the attacker, differential cryptanalysis is not always straightforward. This is because in order to create an input difference during keystream generation, the attacker might need to replay a key-IV pair, which is forbidden by the standard usage protocol for stream ciphers.

2.3.3 Guess-and-determine attacks

Guess-and-determine attacks exploit the relationships between internal values in the stream cipher state, and the relationship between those values and generated keystream. In a guess-and-determine attack, the attacker guesses the value of some component(s), and uses the keystream to determine the values of other components, or to perform consistency checking of previous guessing. The attack succeeds if the attacker can correctly guess the value of a component that is smaller than the size of the key, that in turn leads to the recovery of the entire state.

2.3.4 Correlation and linear masking attacks

Correlation attacks exploit linear relations between internal components and keystream components, or between sequences of keystream; ie. some linear combination of the input and output bits of a non-linear function has a bias towards zero or one.

Some stream ciphers hide biases in the non-linear components using linear masking components. A linear masking attack identifies linear approximations of the non-linear components, and applies it over consecutive steps such that the linear masking cancels, allowing the bias to be observed.

3 Security status of 18033 encryption algorithms

3.1 Asymmetric ciphers - 18033-2

3.1.1 HC

The security status of HC (Hybrid Cipehr) has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This scheme is parameterized in terms of a key encapsulation mechanism KEM and a data encapsulation mechanism DEM. It can be shown that if KEM satisfies the definition of security in Annex B.5 of ISO/IEC 18033-2:2006 and DEM satisfies the definition of security in Annex B.6 of ISO/IEC 18033-2:2006, then HC satisfies the definition of security in Annex B.4 of ISO/IEC 18033-2:2006.

More specifically, for any HC $[t, q, l, l']$-adversary A, we have

$$Adv_{HC}(A)?2\dot{A}dv_{KEM}(A_1) + Adv_{DEM}(A_2).$$

where

- A_1 is a $KEM[t_1, q]-$adversary, with $t_1 \approx t$, and
- A_2 is a $DEM[t_2, q, l, l']-$adversary, with $t_2 \approx t$.

The above inequality does not take into account the possibility that $KEM.KeyGen$ outputs a gbadh key pair (i.e., one for which the decryption algorithm does not act as the inverse of the encryption algorithm) with non-zero probability. In this case, one must simply add this probability (which is assumed to be negligible) to the right hand side of the above inequality. This bound is easily established from the definitions. See, for example, [18] for a detailed proof in the case where there are no labels. The proof in the case of labels can be established along similar lines of reasoning to that in [18]. If KEM is benignly malleable , then one can easily show that HC is also benignly malleable with the same security bound as above.

3.1.2 ECIES-KEM

The security status of ECIES-KEM has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This clause discusses the security of the key encapsulation mechanism ECIES-KEM. This scheme is parameterized in terms of a concrete group Γ and a key derivation function KDF. This scheme can be shown secure in the random oracle model, where KDF is modeled as a random oracle, assuming the Gap-CDH problem is hard.

More specifically, suppose that the system parameters of ECIES-KEM are selected so that $SingleHashMode = 0$ and

$$CheckMode + CofactorMode + OldCofactorMode > 0.$$

Then if A is a $ECIES-KEM[t, q]$-adversary that makes at most q' random oracle queries, then we have

$$Adv_{ECIES-KEM}(A) = O(AdvGapCDH_{\Gamma}(A')),$$

where

A' is a $GapCDH_{\Gamma}[t', O(q')]$-adversary, where $t \approx t$.

This bound is essentially proved in [18], at least for the case where $CheckMode = 1$ and group elements have unique encodings. The other cases can be proved by similar reasoning.

Alternatively, suppose that the system parameters of ECIES-KEM are selected so that $SingleHashMode = 1$ and

$$CheckMode + CofactorMode + OldCofactorMode > 0.$$

In this case, ECIES-KEM is no longer secure against adaptive chosen ciphertext attacks, but it is benignly malleable. If A is a $ECIES - KEM[t, q]$-adversary that makes at most q' random oracle queries, then we have

$$Adv'_{ECIES-KEM}(A) = O(AdvGapCDH_{\Gamma}(A')),$$

where A' is a $GapCDH_{\Gamma}[t', O(q \cdot q')]$-adversary, $where t' \approx t$.

Besides satisfying only a weaker definition of security, this reduction is not as tight as in the case where $SingleHashMode = 0$. Also, the quality of the reduction degrades even further with $SingleHashMode = 1$ when one considers the multi-plaintext model formally defined in [19], whereas the reduction does not degrade significantly when $SingleHashMode = 0$.

If

$$CheckMode + CofactorMode + OldCofactorMode = 0,$$

then in both of the above estimates, the term

$$AdvGapCDH_{\Gamma}(A'),$$

must be replaced by

$$\nu \cdot AdvGapCDH_{\Gamma}(A').$$

Therefore, this selection of system parameters should only be used when ν is very small. Instead of analyzing ECIES-KEM in terms of the Gap-CDH assumption in the random oracle model, one can analyze it without the use of random oracles, but under a very specific and non-standard assumption. See [20, 21] for details.

3.1.3 PSEC-KEM

The security status of PSEC-KEM has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This clause discusses the security of the key encapsulation mechanism PSEC-KEM. This scheme is parameterized in terms of a concrete group Γ and a key derivation function KDF. This scheme can be proven secure in the random oracle model, viewing KDF as a random oracle, assuming the CDH problem is hard. More specifically, for a given value of the system parameter $SeedLen$, and for any $PSEC-KEM[t,q]$-adversary A that makes at most q' random oracle queries, we have

$$Adv_{PSEC-KEM}(A) = O(AdvCDH_{\gamma}(A) + (q+q')(\mu^{-1} + 2^{SeedLen})),$$

where A' is a $AdvCDH_{\Gamma}[t', O(q+q')]$-adversary, with $t' \approx t$. This bound is proven in [22]. Also, the security does not significantly degrade in the multi-plaintext model formally defined in [19].

3.1.4 ACE-KEM

The security status of ACE-KEM has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This clause discusses the security of the key encapsulation mechanism ACE-KEM. This scheme is parameterized in terms of a concrete group Γ, a key derivation function KDF, and a hash function Hash. This scheme can be proven secure assuming the DDH problem is hard - it is to be emphasized that this proof of security is not in the random oracle model. Instead, some specific, and fairly standard, assumptions are made about KDF and Hash. More specifically, for any $ACE-KEM[t,q]$-adversary A, we have

$$Adv_{ACE-KEM}(A) = O(AdvDDH_{\Gamma}(A_1) + Adv_{Hash}(A_2) + Adv_{KDF}(A_3) + q \cdot \mu^{-1}),$$

where:

– A_1, A_2, A_3 denote adversaries that run in time essentially the same as A.

– $Adv_{Hash}(A_2)$ denotes the probability that an adversary A_2, given encodings $EU1^*$ and $EU2^*$ of two independent, random elements in G, can find encodings $EU1$ and $EU2$ of elements in G, such that $(EU1, EU2) \neq (EU1^*, EU2^*)$, but

$$Hash.eval(EU1||EU2) = Hash.eval(EU1^*||EU2^*).$$

If the group supports multiple encodings, the adversary can choose the format it wants when $EU1^*$ and $EU2^*$ are generated; furthermore, the adversary may choose to use the same or different formats in its choice of $EU1$ and $EU2$; however, $EU1^*$ and $EU2^*$ must be consistent encodings, and the same holds for $EU1$ and $EU2$.

If $CofactorMode = 1$, then the adversary may choose $EU1$ to be an encoding of an element of H that does not necessarily lie in G. Note that this problem is a second-preimage collision problem, which is generally believed to be a much harder problem to solve than the problem of finding an arbitrary pair of colliding inputs.

– $AdvKDF(A_3)$ denotes the advantage that an adversary A_3 has in distinguishing between the following two distributions. Let u_1 and $\tilde{h}$ be independent, random elements of G, and let $EU1$ be an encoding of u_1. Let R be a random octet string of length KeyLen. The first distribution is $(R, EU1)$, and the second is $(KDF(EU1||E'(\text{tildeh}), KeyLen), EU1)$.

The reader is referred to [18] for a detailed proof for the case where $CofactorMode = 0$ and group elements have unique encodings. The proof is easily adapted to handle the other cases as well, making use of the fact that the decryption algorithm checks for consistent encodings. It is also shown in [18] that ACE-KEM is no less secure than ECIES-KEM , in the sense that for any $ACE-KEM[t, q]$-adversary A, there exists a $ECIES-KEM[t', q]$-adversary A' such that $t' \approx t$ and $Adv_{ECIES-KEM}(A') \approx Adv_{ACE-KEM}(A)$. The proof in [18] is only for the case where $CofactorMode = 0$ and group elements have unique encodings. The proof is easily adapted to handle the other cases as well, again making use of the fact that the decryption algorithm checks for consistent encodings. It is also shown in [18] that if KDF is viewed as a random oracle, then the security of ACE-KEM can be proven based on the CDH assumption. However, this security reduction is not very tight. The proof in [18] is only for the case where CofactorMode = 0 and group elements have unique encodings. The proof is easily adapted to handle the other cases as well. As pointed out in 18033-2, care should be taken in the implementation of ACE-KEM.Decrypt. Specifically, the implementation of ACE-KEM. Decrypt should not reveal the cause of the error in Step g. If an attacker can obtain such information from a decryption oracle, the proof of security under the DDH assumption will no longer be valid; however, even if such information is available, no attack on the scheme is known, and moreover, the scheme is still no less secure than ECIES-KEM .

3.1.5 RSA-ES

The security status of RSA-ES has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This clause discusses the security of the bounded-plaintext-length asymmetric cipher RSAES. The paper [1] analyzes a more general setting in which (a minor variant of) the RSA encoding mechanism $REM1$ is applied to a general "one-way trapdoor permutation," rather than to a specific function such as the RSA function. The analysis is done in the random oracle model, where the key derivation and hash functions are modeled as random oracles. It is proven in [1] that the resulting scheme satisfies a technical property called "plaintext awareness," assuming the underlying permutation is indeed one way. However, as pointed out in [23], plaintext awareness does not imply security against adaptive chosen ciphertext attack - it only implies a weaker notion of security, namely, security against "lunchtime" attacks. Moreover, it is proven in [1] that $REM1$ will in general not yield a cipher that is secure against adaptive chosen ciphertext attack, if the underlying permutation is arbitrary. This negative result does not imply that RSAES is insecure against adaptive chosen ciphertext attack - it only implies that the analysis in [1] does not establish this. In [1], it is shown that RSAES is secure if the encryption exponent e is very small (e.g., $e = 3$). This result was generalized in [24] to general encryption exponent. It should be pointed out, however, that the security reduction in [24] is not very tight - indeed, it is so bad that it actually says nothing at all about the security of RSAES for RSA moduli of up to several thousand bits. The security reduction in [1] for small encryption exponent is significantly better, but still is not quite as tight as one would like. As pointed out in 18033-2, care must be taken in the implementation of RSAES. Specifically, it is essential that the implementation of $REM1.Decode$ should not reveal the cause of the error in Step k; if an attacker can obtain such information from a decryption oracle, then the scheme can be easily broken, as described in [25].

3.1.6 RSA-KEM

The security status of RSA-KEM has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

This clause discusses the security of the key encapsulation mechanism RSA-KEM. This scheme can be easily shown to be secure in the random oracle model, where the system parameter KDF is modeled as a random oracle, assuming the RSA inversion problem is hard. More specifically, for any RSA key generation algorithm $RSAKeyGen$, such that the output (n, e, d) always satisfies $n \geq nBound$, and for any $RSA - KEM[t, q]$-adversary A, we have

$$Adv_{RSA-KEM}(A) \leq Adv_{RSAKeyGen}(A') + q/nBound,$$

where

– A' is a $RSAKeyGen[t']$-adversary, with $t' \approx t$. This inequality does not take into account the possibility that $RSAKeyGen$ outputs a "bad" RSA key with non-zero probability. In this case, one must simply add this probability (which is assumed to be negligible) to the right hand side of the above inequality. For a proof, see [22].

This security reduction is quite tight, unlike those for RSAES. Moreover, in the multi-plaintext model formally defined in [21], the security of RSA-KEM does not degrade at all, due to the random self-reducibility of the RSA inversion problem. In contrast, the security of RSAES degrades linearly in the number of plaintexts, as the random self-reducibility property unfortunately cannot be exploited in this context. Also, unlike RSAES, RSA-KEM does not seem to be susceptible to "implementation" attacks, such as the attack in [25].

3.1.7 HIME(R)

The security status of HIME(R) has not been changed from the publication date of ISO/IEC 18033-2:2006. The description of the security is written in 18033-2 is as follows.

It can be shown that in the random oracle model, where the functions $Hash$ and KDF in $HEM1$ are modeled as random oracles, that HIME(R) is secure against adaptive chosen ciphertext attack, assuming that it is computationally infeasible to factor integers of the form output by algorithm $HIMEKeyGen$. For details, see [26, 17] - note that [17] corrects several mistakes in [26].

3.2 Block ciphers - 18033-3

3.2.1 TDEA

TDEA is a Feistel block cipher that iterates the Data Encryption Standard function three times with 56-bit keys K1, K2, K3, such that $C = E_{K3}(D_{K2}(E_{K1}(P)))$, where E_k represents encryption with key k, and D_k represents decryption with key k. There are two key lengths permitted by ISO/IEC 18033-3; 192-bits, in which K1, K2 and K3 are treated as separate entities, and 128-bits, in which K3 is set to the same value as K1.

In the three key model, a related key attack where $K1' = K1oplusDelta$, $K2' = K2$ and $K3' = K3$ allows the key K1 to be independently brute-forced. A meet-in-the-middle attack on the remainder (two-key double-DES using K2 and K2) can be applied with one chosen related-key query, one chosen-ciphertext query and at most 2^{72} offline trial encryptions citeKSW96. For most applications, this attack will not be feasible due to the strong assumptions about choosing related-keys and ciphertexts.

An advanced meet in the middle attack applies to three-key triple DES with 2^{32} known plaintexts, 2^{113} operations and 2^{88} memory. citeLuc98. Two-key triple DES is susceptible to a chosen-plaintext attack that requires 2^{56} operations and 2^{56} words of memory citeMH81.

The status of TEA with three keys remains as secure, albeit with a security level of 112 bits, compared to the key size of 192 bits. Users who require strong security may wish to avoid TEA with two keys.

3.2.2 MISTY1

The MISTY1 cipher is a Feistel cipher that operates on a 64-bit block using a 128-bit key. The round function FO is iterated eight times. An additional linear FL function is iterated twice after every two rounds, and an additional two times at the start of the block function.

The most threatening single-key attack uses the higher-order differential technique to attack seven rounds of MISTY1 with computation complexity $2^{120.7}$ and data $2^{59.1}$ citeTSS+10.

There are two attacks on full MISTY1, assuming the use of weak keys. A related key differential attack on the full MISTY1 (including FL functions) has a time complexity of $2^{90.93}$ encryptions, and requires data of 2^{61} chosen ciphertexts, using any of $2^{103.57}$ weak keys. A related-key amplified boomerang attack has a complexity of $2^{87.33}$ encryptions, and requires data of $2^{60.5}$ chosen plaintexts using any of 2^{92} weak keys citeLYW13.

The status of MISTY-1 remains as secure.

3.2.3 CAST-128

CAST-128 is a Feistel cipher that operates on a 64-bit block. The native specification uses a key of between five and sixteen bytes, although ISO/IEC 18033-3 constrains the key to a size of sixteen bytes. The round function is iterated sixteen times to perform an encryption or decryption.

There are no significant attacks on CAST-128. The most effective attack is the linear cryptanalysis on CAST-128 reduced to 6 rounds, using the linear cryptanalytic technique with $2^{53.96}$ known plaintexts, and $2^{88.51}$ 6-round encryptions citeWWH09. Wang, Wang, Chow and Hui use a 6-round differential with probability 2^{-53} to attack 9-rounds of CAST-128. However, this only applies to $2^{-23.8}$ of the key space citeWWC+10

There are no specific attacks on full CAST-128, which remains secure.

3.2.4 HIGHT

HIGHT is a lightweight block cipher that processes 64-bit blocks using 128-bit keys. It iterates a round function 32 times. This cipher provides low-resource hardware implementation which is proper to resource constraint environments.

Lu citeLu07 suggested that the safety margin of HIGHT was four rounds, by attacking twenty-eight rounds of HIGHT using 19-round related-key impossible differentials. The extent of the attack was improved to thirty-one rounds by Ozen [OVT+09], leaving no security margin. The feasibility of related-key attacks is limited in many applications.

Round	Attack Type	Complexity		References	note
		Data	Time		
8/16	Differential (chosen plaintext)	2^{125}	2^{122}	Differential cryptanalysis of eight-round SEED	Information Processing Letters 2011 [64]
7/16	Differential	2^{125}	2^{126}	Differential cryptanalysis of a reduced-round SEED	SCN 2002 [65]

Table 1: The summary of the known attacks on HIGHT (CP : Chosen Plaintext, MA : Memory Accesses)

Ozen et al. also presented a 26-round impossible differential attack on HIGHT, which was improved to twenty-seven of thirty-two rounds by Chen, Wang and Preneel citeCWP12.

In 2010, Koo, Hong and Kwon used a related-key boomerang attack with weak keys (representing one quarter of the key space) to attack full-round HIGHT with time complexity $2^{123.169}$, $2^{57.84}$ data and 4 related keys citeKHK10 .

In 2011, the same team presented the first single-key attack on full HIGHT, using biclique cryptanalysis with an 8-round biclique citeHKK11. The computational complexity is $2^{126.4}$ operations, and the attack recovers the secret key. This represents a technical, if not practical break of the cipher. The attack was slightly improved by Song, Lee and Lee citeSLL13 using a 9-round biclique to improve the computation complexity to $2^{125.9}$ operations. The best known single-key attack mounts a key recovery attack on full HIGHT with 2126.4 encryptions and 248 data, and the best known non-single key attack recovers the 128-bit secret key of full HIGHT with 2123.17 encryptions, 257.84 data and 4 related keys.

Some users may wish to use HIGHT only where less than 128 bits of security are required.

See Table 5 for the summary of the known attacks on HIGHT.

3.2.5 AES

The Advanced Encryption Standard (AES) is a Substitution-Permutation Network-based block cipher with a block size of 128 bits and a key of 128, 192 or 256-bits. The round function is iterated 10 times for a 128-bit key, 12 times for a 192-bit key, and 14 times for a 256-bit key.

The first attack on full-round AES applies to its use with a 192-bit (AES-192) and 256-bit (AES-256) key. This is a related-key boomerang attack that applies to AES-256 with time and data complexity of 2^{119} and 2^{77} memory, and four related keys. It works for all 256-bit keys (60 bits of the key are recovered by the attack; the remainder can be recovered by brute force, and optimization techniques). The attack also applies to AES-192 with time complexity 2^{175} and data complexity 2^{123} citeBK09. Care should be taken with key management in order to avoid these high complexity attacks.

Biclique analysis is the first successful single-key attack on AES, recovering all standardized key sizes with operational complexity less than brute force. For AES-128, the complexity is $2^{126.1}$ with 2^8 memory using 2^{88} data and a 3 round biclique; for AES-192, the complexity is $2^{189.74}$ with 2^8 memory using 2^{80} data and a 4 round biclique; for AES-256, the complexity is $2^{254.42}$ with 2^8 memory using 2^{40} data and a 4 round biclique citeBKR11. These attacks are roughly four times more efficient than brute-force attacks, but with the requirement of significantly more data.

The status of the AES remains as secure.

3.2.6 CAMELLIA

Camellia is a Feistel-based block cipher with a block size of 128 bits. Keys of 128, 192 and 256 bits are permissible. We respectively term these versions of Camellia - Camellia-128, Camellia-192, and Camellia-256. The round function is iterated 16 times for a 128-bit key, and 24 times for a 192- or 256-bit key. There are key-dependent linear operations (termed FL and FL^{-1}) inserted after the sixth, twelve (, and eighteenth) rounds. We only consider attacks that incorporate these functions? analysis that does not take into account the FL functions is generally able to extend the attack by a few rounds, but applies to a Camellia variant that is not part of the ISO standard.

The most successful attack technique is impossible differential cryptanalysis, which manages to attack eleven out of sixteen rounds of Camellia-128, twelve out of twenty-four rounds of Camellia 192, and fourteen out of twenty-four rounds of Camellia-256, using chosen plaintexts and nearly the entire code-book citeLLG12.

The status of Camellia remains as secure.

3.2.7 SEED

SEED is a Feistel cipher that processes 128-bit blocks using 128-bit keys. It iterates a round function sixteen times.

Lu, Yap, Henricksen, and Heng citeLYHH13 show that nine of the sixteen rounds can be attacked using a seven round differential. The time complexity of this attack is $2^{126.33}$ operations, with memory of $2^{69.71}$ bytes. The data complexity is nearly the entire codebook. This suggests that SEED is currently very secure.

So far, the best known single key attack of SEED is differential cryptanalysis of eight-round reduced SEED (the number of its full rounds is 16) which is faster than the exhaustive search key [64]. This attack requires about 2125 chosen plaintexts and 2122 eight-round encryptions. See Table **??** for the summary of the known attacks on SEED. We conclude that SEED is secure against differential attack and we have not had any practical attack on SEED until today.

Round	Attack Type	Complexity		References	note
		Data	Time		
8/16	Differential (chosen plaintext)	2^{125}	2^{122}	Differential cryptanalysis of eight-round SEED	Information Processing Letters 2011 [64]
7/16	Differential	2^{125}	2^{126}	Differential cryptanalysis of a reduced-round SEED	SCN 2002 [65]

Table 2: The summary of the known attacks on SEED

3.3 Stream ciphers - 18033-4

3.3.1 MULTI-S01

MULTI-S01 is a mixing function that uses a pseudo-number keystream generator, such as PANAMA, in conjunction with a 256-bit key to generate authenticated ciphertext. It achieves message secrecy and message authentication simultaneously.

The security of MULTI-S01 is based on the underlying keystream generator. There have been no problems reported with PANAMA, so MULTI-S01 is also believed to be secure [2], although Dawson et al. [4] note that there is no proof of security for PANAMA.

Iwata [5] notes that the MULTI-S01 authors' definition of security - which applies only to an attacker in possession of a single known plaintext pair or single known ciphertext - is not sufficiently stringent, although he verifies that their security claims are true. He also indicates that a more substantial security proof for robustness against adaptive ciphertext attacks can be made, although the proof is not provided.

Dawson et al. [4] claim that given the key a MULTI-S01 instance, it is simple to find two messages that produce the same integrity check, which they claim is a serious flaw, but Rogaway [11] disagrees with that appraisal.

Differential cryptanalysis can be used to successfully attack MULTI-S01 if different messages are encrypted under the same key. However, it is a requirement for all stream ciphers that keys are managed correctly. For MULTI-S01, the random number string number Q must always be unique.

Under normal assumptions, the best known attack on MULTI-S01 [10] has a probability of success of at most $(m-1)/(2^n-1)$, where the attacker knowns the plaintext corresponding to an m-block ciphertext, where each block comprises n-bits . This is the authors' self-evaluation.

3.3.2 MUGI

MUGI is a software-oriented stream cipher, based upon the PANAMA pseudo-random number generator, that uses a 128-bit secret key to provide message secrecy. Its design incorporates well-analysed AES SubBytes and MixColumn components. It is more amenable to analysis with block cipher techniques than

PANAMA.

Some evaluators for the CRYPTREC project [2] pointed out that there are design issues with the MUGI cipher, such that it is strongly weakened if small design changes are made. As it stands, there are no reported security problems with MUGI that permit recovery of the key, internal state, or prediction of keystream with complexity less than that of brute-force guessing the key.

3.3.3 SNOW 2.0

SNOW 2.0 is a software-oriented stream cipher constructed from a word-based Linear Feedback Shift Register coupled with a small Finite State Machine. It uses a 128- or 256-bit key to provide message secrecy.

Watanabe, Biryukov and De Canniere [12] proposed an application of Coppersmith's linear masking to SNOW 2.0. This is a key recovery attack with a complexity of $O(2^{224})$, which is better than brute-force, but impractical due to its high time complexity and requirement of 2^{230} bits of keystream.

Lee, Lee and Park [7] used linear masks to mount a correlation attack on SNOW 2.0 with a time complexity of $O(2^{204.38})$ and data complexity of $O(2^{198.77})$ bits.

3.3.4 Rabbit

Rabbit is a software-oriented stream cipher that uses a 128-bit key and 64-bit IV to provide message secrecy.

There is a distinguishing attack by Lu et al. against the cipher with bias 2^{158} [8], which has been improved to 2^{136} in [9]. Neither attack recovers the key, or can be applied in less time than it takes to derive the key using brute force.

3.3.5 Decim v2

Decim v2 is a hardware-oriented stream cipher with a state-size of 192 bits, that takes a 80-bit secret key and 64-bit IV to provide message secrecy.

While there was an attack on Decim v1 [44], there are no known attacks on Decim v2.

3.3.6 KCipher-2

K2 is a software-based stream cipher that takes a key of 128- or 256- bits, along with a equal sized IV, to provide message secrecy. The structure of the cipher incorporates two shift registers, one driven by the other through a 'Dynamic Feedback Control' mechanism, coupled with a 128-bit Finite State Machine.

There are no known specific attacks that apply to unmodified versions of K2.

Components	Resources Utilized (Slices)	Freq (MHz) (f)	Clock Cycle (c)	Latency msec (c/f)	Throughput (per sec) (f/c)
Encryption (secp256k1)	14,300	42	52,000	1.3	769
Decryption (secp256k1)	14,300	42	52,000	1.3	769
Encryption (secp256r1)	13,700	43.5	80,000	1.9	526
Decryption (secp256r1)	13,700	43.5	80,000	1.9	526
Encryption (sect283r1)	44,807	38.4	4, 900	0.13	7,812
Encryption (sect283r1)	44,807	38.4	4, 900	0.13	7,812

Table 3: Area and Timing for PSEC-KEM on Xilinx Virtex V FPGA

4 Implementation status of 18033 encryption algorithms

4.1 Hardware performance

4.1.1 Asymmetric ciphers - 18033-2

HC There is no explicit published hardware implementation result for HC.

ECIES-KEM There is no explicit published hardware implementation result for ECIES-KEM.

PSEC-KEM In [15], hardware implementation result is shown. It contains area and timing report of PSEC-KEM hardwares on Xilinx Virtex V platform. Both secp256k1 and secp256r1 use almost similar design. The only difference is in the scalar multiplication and modular reduction unit. No DSP blocks have been used to design any primitive. All integer operations are performed using LUT based prime field primitives. The results are taken from Xilinx ISE (Version 11.1) post-rout report. Area and timing reports are presented in Table 3.

The table also contains implementation results for sect283r1 (binary field) and secp256r1 (prime field without using endomorphism). It can be seen that PSEC-KEM for elliptic curves with efficiently computable endomorphism is around 31 % faster compared to generic curves over prime fields on hardware platforms. However, secp256k1 is still slower compared to sect283r1. This happens due to delay involved due to carry propagation in prime field arithmetic. Further improvement in computation time can be achieved by incorporating DSP blocks to design prime field primitives and using pre-computation based window methods for scalar multiplication.

ACE-KEM There is no explicit published hardware implementation result for ACE-KEM.

RSA-ES In [16], hardware implementation results are shown.

Dedicated hardware: For NetSwift-1000, 1024bit key-size and $e = 3$, encryption is done in 1ms and decryption with CRT is done in 26ms. and for 2048bit key-size, $e = 3$, encryption is done in 2ms and decryption with CRT is done in 25ms.

For IBM 4758 (002/003) crypto coprocessor 1024bit key-size and $e = 3$, encryption is done in 2ms and decryption with CRT is done in 6ms.

Samrt card coprocessor: For SGS-Thomson ST19KF16 with 10 MHz, 1024bit key-size and $e = 2^{16} + 1$, encryption is done in 5ms and decryption with CRT is done in 110ms. and for 2048bit key-size, encryption is done in 100ms and decryption with CRT is done in 780ms.

For Philips P83W8532 ST19KF16 with 5 MHz, 1024bit key-size and $e = 2^{16} + 1$, encryption is done in 25ms and decryption with CRT is done in 160ms. and for 2048bit key-size, encryption is done in 54ms and decryption with CRT is done in 1100ms.

For Siemens SLE66CX160S with 5 MHz, 1024bit key-size and $e = 2^{16} + 1$, encryption is done in 24ms and decryption with CRT is done in 230ms. and for 2048bit key-size, encryption is done in 268ms and decryption with CRT is done in 1475ms.

For NEC PD789828 with 40 MHz, 1024bit key-size and $e = 2^{16} + 1$, encryption is done in 7ms and decryption with CRT is done in 100ms. and for 2048bit key-size, encryption is done in 45ms and decryption with CRT is done in 750ms.

RSA-KEM There is no explicit published hardware implementation result for RSA-KEM.

HIME(R) According to self evaluation document for CRYPTREC project [17], HIME(R) can be implemented by 153.7 K gates. With 33MHz clock, encryption is done in 103 μs and decryption is done in 49.4 ms.

4.1.2 Block ciphers - 18033-3

TDEA In hardware, Triple DES may run at 13.3 Gb/s when clocked at 207 MHz [45].

MISTY-1 MISTY-1 runs at 197 Mbps when clocked at 92.6 MHz [46].

CAST-128 CAST-128 hardware using circuits of 26.4-39.5 Kgates obtains performance of 189.9-614.7 Mbps [47].

HIGHT HIGHT is a lightweight software appropriate for hardware. It obtains 188.2 Kb/s at 100 Hz using 3048 GE [48].

AES AES runs at 80 Kb/s on a chip clocked at 100 Hz using 3100 GE [49].

Camellia Camellia uses 321 slices to obtain a throughput of 32.96 Mbps based on Xilinx Spartan-S XC3S50-5 chip and 4.31K gates with a throughput of 81 Mbps based on 0.13-nm CMOS standard cell library [50].

Seed On a Virtex V FPGA, Seed can obtain 6.4 Gbps [51].

4.1.3 Stream ciphers - 18033-4

MULTI-S01 In their self-evaluation of the cipher, the authors of MULTI-S01 describe an implementation with 0.35 nm CMOS, using 140K gates that achieves 9.1 Gbps [63].

MUGI In FPGA at frequency of 95 MHZ, MUGI can reach a throughout of 6.08 Gbps [62].

SNOW 2.0 In FPGA at frequency of 141 MHZ, SNOW 2.0 can reach a throughout of 4.51 Gbps [62].

Rabbit The authors of Rabbit, which is a software based cipher, estimate that in ASIC hardware, the cipher performs at 12.4 GBit/s for 8 pipelines comprising 100K Gates. For Xilin Spartan 3 FPGA with a 2 pipeline design, the authors estimate performance of 8.9 Gbits/s [61].

Decim In FPGA, Decim has a throughput of 165 Mbps running at frequency of 165.73 MHz, using 648 ALUTs and 330 Registers [60].

KCipher2 The authors of K2 consider four implementation on Xilinx Virtex-5 FPGA with the highest performing at 15.2 Gb/s [59].

4.2 Software performance

4.2.1 Asymmetric ciphers - 18033-2

HC There is no explicit published software implementation result for HC.

ECIES-KEM There is no explicit published software implementation result for ECIES-KEM.

PSEC-KEM In [15], software implementation results are shown. This implementation is done on Intel Core 2 Duo, 2.00 GHz processor. Operating system used is Ubuntu 10.04, 32 bit. Compiler is gcc 4.4.3. Encryption and decryption operations using affine coordinate system take around 18 msec for secp256r1. For secp256k1, encryption and decryption operations for PSEC-KEM using affine coordinate system take around 14 msec.

ACE-KEM There is no explicit published software implementation result for ACE-KEM.

RSA-ES In [16], software implementation results are shown. This implementation is done on Celeron 450 MHz, and C++ language. For 1024bit key-size, $e = 3$, encryption is done in 1ms and decryption with CRT is done in 27ms. For 2048bit key-size, $e = 3$, encryption is done in 2ms and decryption with CRT is done in 183ms.

RSA-KEM There is no explicit published software implementation result for RSA-KEM.

HIME(R) According to self evaluation document for CRYPTREC project [17], the implementation result on Pentium III 800MHz, 256MB RAM, Microsoft Windows 98, and ANSI C is 0.6 ms for encryption and 37ms for decryption.

4.2.2 Block ciphers - 18033-3

4.2.3 TDEA

Triple-DES is better optimized for hardware. It runs at 108 cycles/byte on a Pentium [52].

4.2.4 MISTY-1

On a Pentium II, MISTY-1 runs at 37.5 cycles/byte [53].

4.2.5 CAST-128

CAST-128 in counter mode runs at 31.9 cycles/byte on for C++ code on an Intel Core 2 1.83 GHz machine [54].

4.2.6 HIGHT

HIGHT is a lightweight software appropriate for hardware. Software speed is unknown [55].

4.2.7 AES

AES has native instruction support on recent Intel x86 architectures. In GCM mode, this permits AES to run at about 3.5 cycles/byte [56].

4.2.8 Camellia

Camellia runs at 36.3 cycles per byte for C++ code run on an Intel Core 2 1.83 GHz [57].

4.2.9 Seed

Seed runs at 59.2 cycles per byte for C++ code run on an Intel Core 2 1.83 GHz [58].

4.2.10 Stream ciphers - 18033-4

The results of software implementation of stream cipher are indicated Table 4, 5, and 6.

Processor	Speed (MHz)	Cipher					
		Decim	MUGI	MULTI-S01	Rabbit	SNOW 2.0	K2
Intel Pentium M [3]	1700	12975			3.94	4.75	
Intel Pentium 4 [3]	2800	12845			9.46	5.04 / 5.01 (1)	
Intel Pentium 4 [3]	3000	8347 (2)			7.71	5.19 / 5.22 (1)	
Intel Pentium 4 [6]	3200						5.4
AMD Athlon 64 X2 4200+ [3]	2200	5195			4.98	4.83	
PowerPC G4 [3]	1670	7789 (2)			11.78	7.43	
PowerPC G4 [3]	533	4034 (2)			16.8	7.06	
Alpha EV6 [3]	500	7820			5.27	5.17 / 5.18 (1)	
HP 9000/785 [3]	875	22420			7.2	4.52	
UltraSparc-III [3]	750	20129 (2)			11.6	7.53	
Intel Pentium III	650		6.5 [14]	14.9 [2]			
Intel Pentium III [2]	800		21.8				
Alpha 21164A [2]	600			17.7			

Table 4: Speed of stream cipher encryption(cycles/byte): (1) Refers to encryption using 128-bit and 256-bit keys respectively (2) Decim version not denoted - possibly Decim v1.

Processor	Speed (MHz)	Cipher					
		Decim	MUGI	MULTI-S01	Rabbit	SNOW 2.0	K2
Intel Pentium M [3]	1700	18			549	77/90 (1)	
Intel Pentium 4 [3]	2800	55.6			984	126/85 (1)	
Intel Pentium 4 [3]	3000	45 (2)			618	90/140 (1)	
Intel Pentium 4 [6]	3200						1136
AMD Athlon 64 X2 4200+ [3]	2200	12			288	43/70 (1)	
PowerPC G4 [3]	1670	36.2 (2)			717	88/136 (1)	
PowerPC G4 [3]	533	35.8 (2)			1038	74/108 (1)	
Alpha EV6 [3]	500	31.2			5.27	691/105 (1)	
HP 9000/785 [3]	875	76			526	109/173 (1)	
UltraSparc-III [3]	750	106.5 (2)			605	88/128 (1)	
Intel Pentium III	650		9187/9967 [14]	5649 [2]			
Intel Pentium III [2]	800		15029				
Alpha 21164A [2]	600			31737			

Table 5: Cycles required for key initialization: (1) Refers to encryption using 128-bit and 256-bit keys respectively (2) Decim version not denoted - possibly Decim v1.

Processor	Speed (MHz)	Cipher					
		Decim	MUGI	MULTI-S01	Rabbit	SNOW 2.0	K2
Intel Pentium M [3]	1700	338913			455	746/769 (1)	
Intel Pentium 4 [3]	2800	359916			826	1029 / 1001 (1)	
Intel Pentium 4 [3]	3000	132752 (2)			7.71	1209/1082 (1)	
AMD Athlon 64 X2 4200+ [3]	2200	111732			4.98	528/541 (1)	
PowerPC G4 [3]	1670	110272 (2)			11.78	644/670 (1)	
PowerPC G4 [3]	533	57200 (2)			16.8	704/720 (1)	
Alpha EV6 [3]	500	173660			5.27	489/492 (1)	
HP 9000/785 [3]	875	547883			7.2	469/479 (1)	
UltraSparc-III [3]	750	291750 (2)			11.6	742/769(1)	

Table 6: Cycles required for IV initialization

References

[1] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," In Advances in Cryptology, Eurocrypt 94, pp.92-111, 1994.

[2] Information Technology Promotion Agency, Japan; Telecommunications Advancement Organization, Japan. CRYPTREC report 2002. Available at: http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/c02e_report2.pdf

[3] ESTREAM Project. "Latest Performance Figures", Available at http://www.ecrypt.eu.org/stream/perf/#results

[4] E. Dawson, A. Clark, H. Gustafson, B. Millan and G.Carter, "Evaluation of MULTI-S01", Report submitted to CRYPTREC, 2002. Available at http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1088_Analysis_of_MULTI-SO1.pdf

[5] T. Iwata, "Security Evaluation Report on MULTI-S01", Report submitted to CRYPTREC, 2002. Available at http://www.cryptrec.go.jp/estimation/rep_ID0030.pdf

[6] S. Kiyomoto, T. Tanaka and K. Sakurai, "K2: A Stream Cipher Algorithm using Dynamic Feedback Control", Proceedings of SECRYPT 2007, 2007, pages 204-213.

[7] J. K. Lee, D. H. Lee and S. Park, "Cryptanalysis of Sosemanuk and SNOW 2.0 Using Linear Masks", Proceedings of ASIACRYPT 2008, pages 524-538.

[8] Y. Lu, H. Wang, and S. Ling, "Cryptanalysis of Rabbit", Proceedings of Information Security Conference, 2008, pages 204-214.

[9] . Y. Lu, S. Vaudenay, W. Meier, L. Ding and J. Jiang. "Synthetic Linear Analysis: Improved Attacks on CubeHash and Rabbit", Proceedings of ICISC 2011, 2011.

[10] C. Mitchell and A. Dent, "International standards for stream ciphers: A progress report", 2004, Available at http://www.chrismitchell.net/isfsca.pdf.

[11] P. Rogaway, "Security Level of Cryptography in Security evaluation (especially in mode part) for the stream cipher MULTI-S01", Report submitted to CRYPTREC, December 2001. Available at http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1038_multi.pdf

[12] D, Watanabe, A. Biryukov, C. De Canniere:, "A Distinguishing Attack of SNOW 2.0 with Linear Masking Method," Selected Areas in Cryptology 2003, pp. 222-233.

[13] H. Wu and B. Preneel. Cryptanalysis of the stream cipher DECIM, in Proceedings of FSE2006, pages 30-40.

[14] H. Yoshida and S. Furuya, "Considerations related to software fast implementation of pseudo-random number generator", 2003 cipher and information security symposium and proceedings of SCIS 2003, 9C-4, 2003.

[15] S. S. Roy and D. Mukhopadhyay, " Implementation of PSEC-KEM (secp256r1 and secp256k1) on Hardware and Software Platforms Final Project Report," `http://cse.iitkgp.ac.in/~debdeep/osscrypto/psec/downloads/PSEC-KEM_prime.pdf`

[16] RSA Laboratories, "RSAES-OAEP Encryption Scheme," `ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf`

[17] "Self-evaluation report: HIME(R) cryptosystem," Oct. 2003, Available from `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`.

[18] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," Cryptology ePrint Archive, Report 2001/108, 2001. `<http://eprint.iacr.org>.`

[19] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," In Advances in Cryptology, CRYPTO 98, pp. 26-45, 1998.

[20] M. Abdalla, M. Bellare, and P. Rogaway, "DHAES: an encryption scheme based on the Diffie-Hellman problem," Cryptology ePrint Archive, Report 1999/007, 1999. `<http://eprint.iacr.org>.`

[21] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," In Topics in Cryptology ? CT-RSA 2001, pages 143?158, 2001. Springer LNCS 2045.

[22] V. Shoup "A proposal for an ISO standard for public key encryption," Cryptology ePrint Archive, Report 2001/112, 2001. `<http://eprint.iacr.org/>.`

[23] V. Shoup "OAEP reconsidered," In Advances in Cryptology?Crypto 2001, pages 239?259, 2001.

[24] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is secure under the RSA assumption," In Advances in Cryptology?Crypto 2001, pages 260?274, 2001.

[25] J. Manger. "A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS # 1 v2.0," In Advances in Cryptology?Crypto 2001, pages 230?238, 2001.

[26] M. Nishioka, H. Satoh, and K. Sakuri, "Design and analysis of fast provably secure public- key cryptosystems based on a modular squaring," In Proc. ICISC 2001, LNCS 2288, pages 81?102, 2001.

[27] A. Biryukov and D. Khovratovich, "Related-Key Cryptanalysis of the Full AES-192 and AES-256", Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009, pages 1-18.

[28] A. Bogdanov, D. Khovratovich and C. Rechberger. "Biclique Cryptanalysis of the Full AES", 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, pages. 344-371.

[29] J. Chen, M. Wang and Preneel, "Impossible Differential Cryptanalysis of the Lightweight Block Ciphers TEA, XTEA and HIGHT", Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrane, Morocco, July 10-12, 2012. Proceedings, pages 117-137.

[30] Y. Liu, L. Li, D. Gu, X. Wang, Z. Liu, J. Chen and W. Li, "New observations on impossible differential cryptanalysis of reduced-round Camellia," FSE 2012.

[31] D. Hong, B. Koo and D. Kwon, "Biclique Attack on the Full HIGHT", Information Security and Cryptology - ICISC 2011 - 14thInternational Conference, Seou l, Korea, November 30 - December 2, 2011., pages 365-374.

[32] B. Koo, D. Hong and D. Kwon, "Related-key attack on the Full HIGHT", Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, pages 49-67.

[33] L. Jiqiang, W. S. Yap, M. Henricksen and H. Swee-Huay, "Differential Attack on Nine Rounds of the SEED", awaiting publication.

[34] J. Lu, "Cryptanalysis of Reduced Versions of the HIGHT Blcok cipher from CHES 2006", Information Security and Cryptology - ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings, pages 11-26.

[35] O. Özen, K. Varici, C. Tezcan and Ç. Kocair, "Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT", Information Security and Privacy, 14th Australasian Conference, ACISP 2009, Brisbane, Australia, July 1-3, 2009, Proceedings, pages 90-107.

[36] J. Song, K. Lee and H. Lee, "Biclique cryptanalysis on lightweight block cipher: HIGHT and Piccolo", International Journal of Computer Mathematics, 2013, pages, 1-17.

[37] Y. Tsunoo, T. Saito, M. Shigeri and T. Kawabata. "Security Analysis of 7-Round MISTY1 against Higher Order Differential Attacks", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol 2010, No 1, 2010, pp 144-152.

[38] J. Lu, W. S. Yap, Y. Wei, "Weak Keys of the Full MISTY1 Block Cipher for Related-Key Cryptanalysis", Topics in Cryptology - CT-RSA 2013, Lecture Notes in Computer Science Volume 7779, 2013, pp 389-404.

[39] S. Lucks, "Attacking Triple Encryption", Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings, pp 239-253.

[40] R. Merkle and M. Hellman, "On the Security of Multiple Encryption", Communications ACM, vol 24, no 7, 1981, pp. 465-467.

[41] J. Kelsey, B. Schneier and D. Wagner. "Key-schedule cryptanalysis of IDEA, gDES, GOST, SAFER and triple-DES," In Advances in Cryptology, CRYPTO 96, pp. 237-251. Springer Berlin Heidelberg, 1996.

[42] M. Wang, X. Wang, K. P. Chow, L. C. K. Hui, "New differential cryptanalytic results for reduced-round CAST-128," Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vE93-A n.12, 2010, p2744-2754.

[43] M. Wang, X. Wang, C. Hu, "New Linear Cryptanalytic Results of reduced-round of CAST-128 and CAST-256," Selected Areas in Cryptography Lecture Notes in Computer Science Volume 5381, 2009, pp 429-441.

[44] H. Wu and B. Preneel. "Cryptanalysis of the stream cipher DECIM," in Proceedings of FSE2006, pages 30-40.

[45] `http://www.xilinx.com/support/documentation/application_notes/xapp270.pdf`

[46] `https://www.iacr.org/archive/ches2008/51540311/51540311.pdf`

[47] T. Sugawara, N. Homma, T. Aoki, A. Satoh, " A High-Performance ASIC Implementation of the 64-bit Block Cipher CAST-128"

[48] `http://www.ecrypt.eu.org/lightweight/index.php/Block_ciphers`

[49] `http://www.ecrypt.eu.org/lightweight/index.php/Block_ciphers`

[50] `http://www.researchgate.net/publication/221131232`
`A_pipelined_camellia_architecture_for_compact_hardware_implementation`

[51] `http://link.springer.com/chapter/10.1007%2F978-3-642-00641-8_19#page-2`

[52] `http://cryptome.org/jya/aescomm.htm`

[53] `http://www.researchgate.net/publication/240269188_Supporting_Document_of_MISTY1`

[54] `http://www.cryptopp.com/benchmarks.html`

[55] `http://www.ecrypt.eu.org/lightweight/index.php/Block_ciphers`

[56] T. Krovetz, W. Dai (2010). "How to get fast AES calls?".Crypto++ user group. Retrieved 2010-08-11.

[57] `http://www.cryptopp.com/benchmarks.html`

[58] `http://www.cryptopp.com/benchmarks.html`

[59] Y. Nakano, K. Fukushima, S. Kiyomoto, and Y. Miyake. Fast Implementation of Stream Cipher K2 on FPGA, http://waset.org/publications/15088/fast-implementation-of-stream-cipher-k2-on-fpga

[60] J. M. Marmolejo-Tejada, V. Trujillo-Olaya, J. Velasco-Medina, Hardware Implementation of Grain-128, Mickey-128, Decim-128 and Trivium, http://www.teleinfo.cefetpr.br/disciplinas/congressos/Latincom2010/ANDESCON/DATA/ADC-01-04.pdf

[61] M. Boesgaard, M. Vesterager, T. Christensen, and E. Zenner, The Stream Cipher Rabbit, http://cr.yp.to/streamciphers/rabbit/desc.pdf

[62] P. Kitsos, Hardware Implementations for the ISO/IEC 18033-4:2005 Standard for Stream Ciphers, World Academy of Science, Engineering and Technology, International Journal of Electrical, Robotics, Electronics and Communications Engineering Vol:1 No:6, 2007, http://waset.org/publications/12965/hardware-implementations-for-the-iso-iec-18033-4-2005-standard-for-stream-ciphers

[63] Hitachi Ltd, Self-Evaluation Report MULTI-S01 (revised for 2001 submittion), http://www.hitachi.com/rd/yrl/crypto/s01/01eeval.pdf

[64] J. Sung, Differential cryptanalysis of eight-round SEED, Information Processing Letters, Vol. 111 No. 10, pp. 474-478, Elsevier, 2011.

[65] H. Yanami and T. Shimoyama, Differential Cryptanalysis of a Reduced-Round SEED, SCN 2002, LNCS 2576, pp. 186-198, Springer, 2003.