Strength Assessment Of Encryption Algorithms

White Paper

October 2000

Discretix Technologies Ltd.

Authors: Limor Elbaz & Hagai Bar-El
Email: {limor.elbaz, hagai.bar-el}
@discretix.com
Tel: +972-9-8858810
www.discretix.com

INTRODUCTION

This paper presents the results of a study undertaken to research some of the most commonly used encryption algorithms firstly to identify as many of their weaknesses and vulnerabilities as possible, and secondly to identify the aspects of these weaknesses that can be avoided by correct implementation.

METHODOLOGY

The research included 30 professional articles, published by various well-known cryptographers. These were reviewed, analyzed and then evaluated for their relevance. The list of articles is provided in the last chapter of this paper.

This paper is intended to be read by engineers and decision-makers, who may not be familiar with advanced terms in cryptography or in mathematics, thus all proofs and raw theorems have been omitted. The reader may however assume that every statement, appearing in this document, regarding a breach in an algorithm, is backed up by published research along with adequate proofs.

This paper contains a list of known weaknesses in algorithms. The relevant weaknesses are presented for each of the examined algorithms, along with their practical meaning as identified by the Discretix Research team.

This document is not meant to give any descriptive information about the various algorithms and therefore does not contain any algorithm-specific description other than that of the identified vulnerabilities. The reader is encouraged to refer to the appropriate specifications for information about the structure of the algorithms that are of his/her interest.

TYPES OF ATTACKS

This section briefly overviews common terminology related to types of attacks. These terms are used throughout the document.

- **Brute-force Attack**: Brute-force is the ultimate attack on a cipher, by which all possible keys are successively tested until the correct one is encountered. A brute-force attack cannot be avoided but it can be made infeasible.
- **Codebook Attacks**: Codebook attacks are attacks that take advantage of the property by which a given block of plaintext is always encrypted to the same block of ciphertext as long as the same key is used. There are several types of codebook attacks. The most typical ones are using character occurrence probabilities in plaintext.
- **Differential Cryptanalysis**: Differential cryptanalysis is the attempt to find similarities between various cipher-texts that are derived from similar (but not identical) plaintexts. This similarity may assist in recovering the key.
- **Known Plaintext Attacks**: These are attacks in which the attacker knows the plaintext as well as the ciphertext of an encryption operation and attempts to recover the key.

- **Linear Cryptanalysis**: Linear cryptanalysis is the attempt to find linear dependency of high probability between the plaintext, the ciphertext and the key, by which the key may be retrieved.
- **Man-In-The-Middle Attack (MIM, or MITM)**: A "man-in-the-middle" attack is an attack that is placed by an active attacker who can listen to the communication between two entities and can also change the contents of this communication. While performing this attack, the attacker pretends to be one of the parties in front of the other party.
- **Oracle Attack**: An Oracle attack is in attack during which the attacker can be assisted by a machine or user who will perform encryption or decryption for him at will. The attacker can use multiple encryptions and decryptions of data of his choice to recover the key.
- **Related-Key Cryptanalysis**: Related-key cryptanalysis refers to attacks based on encrypting plaintexts with various similar (but not identical) keys and analyzing the differences in output.

ALGORITHMS VULNERABILITIES

The following chapter presents the identified weaknesses for each of the evaluated algorithms.

SYMMETRIC BLOCK CIPHERS

The following section presents the weaknesses that were found in algorithms for *symmetric encryption*. A symmetric encryption algorithm is an algorithm by which decryption and encryption are performed using the same key. Such algorithms are often called "*Private key algorithms*". These algorithms, being noticeably faster, are used for bulk encryption.

DES (Data Encryption Standard)

The DES algorithm, other than its short key (which can be *brute-forced* quite easily), is known to be secure. Triple-DES was developed to overcome this limitation.

Chaum and Evertse demonstrated an attack, on six rounds of DES, in a time of 2^{54}. This attack cannot be applied on more than eight rounds (let alone sixteen) so is not alarming in practice.

Davies *known plain-text attack*, using S-box pairs, succeeded in cracking eight rounds of DES using 2^{40} known plaintexts and with 2^{40} operations work. This is not too alarming since it has almost no effect on the strength of the full sixteen-round DES. Extension of this particular attack to sixteen rounds will take more than the entire codebook of plaintexts and is therefore impractical. Yet, an improvement to this attack could crack full-DES (sixteen rounds) with a time of 2^{50} for data collection plus a time of 2^{50} for the attack itself. There is a tradeoff between success rate, time and amount of required plaintexts. The abovementioned figure presents the best "deal" in these terms. Alternatively, the attack can be performed to the extent it reveals 24 bits of the 56-bit key using 2^{52} known plaintexts, with very little work (complexity). These results may be alarming for systems in which bulk data is encrypted with unchanged keys.

The DES algorithm suffers from *Simple Relations* in its keys. In DES, the simple relationship is of a complementary nature. This means that the complementary relationship between keys results in a complementary relationship between the resulting ciphertexts. This vulnerability reduces the algorithm strength by one bit. Other relationships are existent for some other specific keys as well.

With regards to *weak keys*, DES has at least four of them. When encrypting using one of these weak keys, all sixteen rounds will be using the same sub-keys, making the algorithm as strong as a single round. Therefore, use of these keys must be avoided. In addition to these four keys, there are twelve more weak keys by which two rounds are running using the same sub-keys. In addition to these weak keys, DES also has keys that are defined as *weak*[1] and keys that are defined as *semi-weak*[2]. All these keys should be avoided so as not to harm the strength of the implementation when using the algorithm.

The key schedule that DES uses is not one-way. This results in the attacker being able to recover most of the master-key by compromising the sub-keys of few rounds. This vulnerability is hardly practical since the round keys are not easily available. Yet, this feature does assist in optimizing differential attacks.

The DES algorithm is vulnerable to *linear cryptanalysis* attacks. By such an attack, the algorithm in its sixteen rounds can be broken using 2^{43} known plaintexts. This vulnerability raises a notable risk when encrypting bulk data that may be predictable with keys that are constant.

Eli Biham and Adi Shamir presented a differential attack, by which a key can be recovered in 2^{37} time using 2^{37} ciphertexts taken from a pool after encrypting 2^{47} chosen plaintexts, even if these ciphertexts were encrypted with various keys. This attack, although very interesting academically, is hard to mount in most circumstances.

DES has several modes of operation, most commonly used one being CBC. Yet, if such modes (other than ECB) are used, it must be verified that the IV (Initialization Vector) is either fixed or not transferred in the clear. Otherwise, the implementation is highly vulnerable in the existence of an active attacker[3].

The 3DES algorithm, at least theoretically, is vulnerable to *linear* and *differential* attacks.

Triple-DES (3DES), other than being slow, is vulnerable to a variant of a *meet-in-the-middle attack* together with differential related-key attack. We did not find accurate figures for the cost of such attacks.

RC2

RC2 is an algorithm for which little cryptanalysis is available. However, it is known to have two weaknesses.

First, RC2 is vulnerable to differential attacks. An implementation with *r mixing* rounds (including the accompanying *mashing* rounds) will require at most 2^{4r} chosen plaintexts for a differential cryptanalysis attack. Commonly, the RC2 runs with 16 mixing rounds, making this attack less feasible than it may seem.

[1] A key is called "weak" if when using it the encryption function is similar to the decryption function.

[2] A pair of keys is called "semi-weak" if for one key the encryption function acts as the decryption function of the other.

[3] An attacker who can manipulate the data.

Second, the algorithm is vulnerable to a differential related-key attack requiring only 2^{34} chosen plaintexts and one related-key query.

RC4

The RC4 algorithm was not reviewed publicly to the extent of the others. The main weakness in this algorithm is that due to a weak key-mixing phase, 1/256 of the keys belong to a class of weak keys. These keys are detectable. After detection of a key belonging to this class, it is fairly easy to reveal 16 bits of the key with a 13.8% probability. In any implementation of this algorithm, a test to assure these keys are not used must be performed.

RC5

RC5 was not extensively reviewed either. This algorithm is, however, known to suffer from several weak keys. For RC5 running with *r* rounds, each key has a probability of 2^{-10r} of being a weak key. This weakness is not highly risky in practice although the weak keys should be avoided in the implementation.

One attack, demonstrated by the designers of RC6, can break both RC5 and RC6 when running with up to 15 rounds, requiring less time than needed to perform an exhaustive key search.

RC6

RC6 is considered to be a strong algorithm with a fast and easy hardware implementation. It was submitted as an AES candidate and reached the second AES evaluation round. The RC6 has the following (mostly impractical) documented vulnerabilities.

For RC6 with 15 rounds or less, running on input blocks of 128 bits, it has been shown that the resulting ciphertext could be distinguished from a random series of bits. One of the conditions for an encryption algorithm to be secure is that its output resembles a completely random series of bits. Several applications check for randomness of bit streams to indicate strong encryption. Moreover, the writers of the algorithm have shown an attack against RC6 running with up to 15 rounds that is faster than an exhaustive key search. For one class of weak keys, it was shown that full randomness is not accomplished for up to 17 rounds of the algorithm.

For RC6 with 16 rounds, a linear cryptanalysis attack is possible, but requires 2^{119} known plaintexts, which makes this attack quite infeasible.

The RC6 algorithm is robust against differential cryptanalysis, provided that it applies more than 12 rounds.

CMEA (Cellular Message Encryption Algorithm)

The CMEA algorithm has been used for encryption of control messages (and any other messages) in cellular phones. This algorithm is highly criticized for its lack of strength. This algorithm is by far the weakest algorithm that was examined and most of the criticism to which it is subject is justifiable. The following are brief descriptions of the most alarming vulnerabilities that were encountered.

With regards to weak keys in their original definition, every single key of CMEA is a weak key. In other words it may be said that the CMEA function is its own inverse.

Since the CMEA algorithm does not support a CBC[4] mode or anything similar, nor does it support the use of IVs (Initialization Vectors), codebook attacks are feasible and easy to mount. Codebook attacks are one of the most primitive attacks and require hardly any facilities or computational power. Since the algorithm is also used to encrypt dialed digits that can be easily revealed (directly or by using side-information), codebook attacks are made easy. The encryption algorithm does not protect the entire plaintext that is provided as input, but rather, it protects everything except the last bit. The last bit of the plaintext is always changed to its complement, regardless of the key that is used.

An important element of the CMEA algorithm is the T-Box, which does most of the transformation. The T-Box generates many equivalent outputs for various keys. The number of equivalents is very large, making four bits of the key simply meaningless. Therefore, the effective key length of the algorithm is only 60 bits.

The T-Box makes use of a fixed table of substitution, called the "*Cave Table*". This table provides substitution that is not uniform but is skewed. Due to this property, the output of the T-Box is skewed as well, making some values appear more often and some values never appear.

Recovery of the T-box structure for a given key is enough to break the algorithm and the key itself does not need to be revealed at all. Recovery of the complete T-Box (for any block size) using a chosen-plaintext attack requires only 338 chosen plaintexts and very little computation time. Alternatively, if the block size is three bytes, as in most implementations, complete T-box recovery can be done using known-plaintext attacks requiring as little as 40-80 known plaintexts and 2^{32} operations (that can be done in parallel). If the block size is two bytes (as sometimes used), a complete recovery requires only four known plaintexts and 2^{32} operations. Alternately, the same recovery can be done using only two known plaintexts and some unknown ciphertext blocks.

There is absolutely no doubt that the CMEA algorithm is not suitable for any application that requires even the minimal level of security. This algorithm should be implemented only to satisfy specific needs such as interoperability with existing systems, compatibility with existing infrastructures, etc.

Blowfish

Blowfish was written by Bruce Schneier, a well-known cryptographer. The algorithm is designed to resist all known attacks on block ciphers. Blowfish is known for its high security and is available in several common encryption products.

Blowfish has some classes of weak keys. For these weak keys, separate rounds end up using the same round-keys. Keys belonging to these classes can be detected only in reduced-rounds versions of the algorithm and not on the full blowfish. Blowfish is known to successfully make (almost) every bit of the key affect most of the round-keys.

Blowfish is immune against differential related-key attacks because of the fact that every bit of the master key affects many round keys. The round-keys are

[4] The CBC (Cipher Block Chaining) mode is a mode of operation that causes dependency between every encrypted block and a predecessor block, making codebook attacks ineffective.

highly independent, making related-key attacks very difficult or infeasible. Such independence is highly desirable.

Twofish

Twofish was also invented by Bruce Schneier, and was submitted as an AES candidate. The algorithm is considered to be secure after having been reviewed by several respectable cryptographers.

Mirza and Murphy published an attack by which guessing the 64 bits of the key that form the round keys (for 128-bit key encryption) can reveal information about the S-box, which is key-dependent, due to non-uniform distribution of round keys. According to the creators of the algorithm, the loss of entropy due to this attack is only 0.8 bit (for the specific round-key). Moreover, this attack cannot be applied, as it is, to keys of more than 128-bits. The same technique is claimed to apply to DES and to Triple-DES as well, although further experiments were not published to demonstrate this.

Further, Mirza and Murphy claimed that the number of sub-key pairs possible from a 128-bit key is not 2^{128}, as one would expect (so as not to lose entropy) but only 0.632 of that. The designers of the algorithm claim that the number of unique sub-key pairs is 2^{117} but that this quality does not affect the security of the algorithm.

The Twofish algorithm seems to be quite robust against known types of cryptanalysis according to its authors' claims, which have been proven to some limited extent and have not yet proven to be false.

According to the authors, the algorithm is resistant to related key attacks such as the *slide-attack*[5] and the *related key differential attack*. Also, according to its authors, the algorithm does not have any weak keys in the sense that using these specific keys will result in predictable sub-keys or in predictable round outputs. Related-key characteristics are not existent either. The authors, however, state some specific issues that have not as yet been covered by their analysis, mainly the resistance of the algorithm to chosen-key attacks. Possible vulnerability to these attacks may harm the algorithms security when used in some specific implementations, such as a hash function.

CAST

CAST is often perceived as one of the strongest algorithms available, and is deployed in several common encryption applications. CAST is relatively secure and is built to resist most of the known types of attacks. Following are its known weaknesses.

Whereas CAST is known to be quite resistant against linear cryptanalysis, it's key can be recovered by linear cryptanalysis using a known-plaintext attack. Such an attack on CAST-256 (with 256-bit key) requires 2^{122} known plaintexts (for 48 rounds). This attack is definitely not feasible even when considering only the amount of time it is likely to take. On 8-round CAST, linear cryptanalysis will require only 2^{34} known plaintexts. On 12-round CAST, such an attack will require 2^{50} known plaintexts, which is infeasible due to space constraints (requiring more than 9,000 terabytes of storage space). Similar infeasibility applies for 16-round CAST, which requires 2^{66} known plaintexts.

[5] The Sliding-Attack is an attack by which the opponent shifts rounds forward by key manipulation.

The 64-bit key version of CAST is somewhat vulnerable to *differential related-key cryptanalysis*. It can be broken by 2^{17} chosen plaintexts along with one related-key query in offline work of 2^{48}. This result may be alarming if the implementation of the algorithm gives the attacker the chance to feed in ample amounts of chosen-plaintexts to be encrypted with keys that differ by values that are known to the attacker. However, it is not likely that CAST with 64-bit keys will ever be implemented.

Regarding differential-cryptanalysis, CAST-256 is considered secure. This is, of course, only true when the correct (specified) number of rounds is used. Differential cryptanalysis attack on CAST-256 when employing 48 rounds requires 2^{140} chosen plaintexts, which is clearly infeasible.

Rijndael

Rijndael was the NIST finalist in the AES competition and was declared the new standard symmetric cipher that is expected to replace DES. Briefly, it may be said that it was possible to break only up to eight rounds of the cipher with less work than of an exhaustive key-search (both for 192 and 256-bit keys). Nine rounds can be attacked using related-key attacks, but this is still impractical.

Four attacks were discovered to work against reduced-rounds versions of Rijndael: Square Attack, Improved Square Attack, Impossible Differential Attack and Reversed Key Schedule Attack.

A *Square Attack* breaks four rounds of Rijndael in 2^9 time, requiring 2^9 chosen plaintexts. An *Improved Square Attack* does that the same in 2^8 time. The same *Square Attack* when running on 5-rounds requires 2^{40} time, with 2^{11} chosen plaintexts. The *Improved Square Attack* does the same in 2^{39} time. The *Sqaure Attack* on six rounds requires 2^{72} time using 2^{32} chosen plaintexts while the *Improved Square Attack* will take 2^{71} time. The *Impossible Differential Attack* handles five rounds in 2^{31} time using $2^{29.5}$ chosen plaintexts, whereas a *Reversed Key Schedule attack* requires only 2^{11} chosen plaintexts for the same job. An attack on six rounds can be done using the *Reversed Key Schedule Attack* in 2^{63} time using 2^{32} known plaintexts. Attacks on six rounds were also shown using $6*2^{32}$ known plaintexts in 2^{44} operations.

For seven rounds of Rijndael, attacks on a 192-bit key were shown using $19*2^{32}$ known plaintexts in 2^{155} time. For a 256-bit key this attack would require $21*2^{32}$ known plaintexts and 2^{172} time. These attacks are, of course, infeasible due to length of time required and due to the fact that Rijndael employs more than seven rounds when in practical use.

Attacks on eight rounds will take the entire codebook of known plaintexts and 2^{188} time for a 192-bit key (2^{204} time for a 256-bit key) and are therefore totally inapplicable. A related-key attack on nine rounds of Rijndael with 256-bit key will require 2^{77} plaintexts to be encrypted using 256 related keys and 2^{224} time. This attack is clearly infeasible.

IDEA (IPES)

IDEA is an algorithm that uses a technique of multiple-group operations to gain its strength and to defeat most common attacks. Some vulnerabilities were found for reduced-rounds versions of the algorithm, and several classes of weak keys were also detected.

A successful differential attack was presented for 2.5 rounds[6] of IDEA, requiring 2^{10} chosen plaintexts and 2^{32} time (one day on a standard PC). This attack is believed to work for 3.5 rounds of IDEA in a time that is shorter than the time required for an exhaustive search. It is quite definite, though, that this attack cannot apply to eight rounds of IDEA. The attack is not based in any way on IDEA's key schedule, so it will not be made infeasible even if the key schedule mechanism is improved. Yet, it is not considered as a real threat since it works only on versions that are significantly reduced.

Another attack produced by Borst, Knudsen and Rijmen on reduced IDEA (3 rounds) required 2^{29} chosen plaintext pairs and 2^{44} encryptions. A third attack recovered the key of 3.5 rounds with a probability of 86%, using 2^{56} chosen plaintexts and 2^{67} encryptions. A truncated differential attack on 3.5 rounds of IDEA was found to recover the key using 2^{32} words of memory, 2^{56} chosen plaintexts and 2^{67} operations with a probability of greater than 83%. However, all these attacks, as the first one, do not apply for the full 8.5 rounds algorithm. According to Borst, Knudsen and Rijmen, the attacks can be extended to more than 3 rounds, though it is not likely that 8.5 rounds (full) IDEA is at any risk.

Full IDEA has 8 rounds, but the first 3 rounds seem to be highly vulnerable to related-key attacks such as key-schedule attacks and related-key differential timing attacks (by comparing the time it takes to perform decryption using multiple related keys), in addition to the attacks on reduced rounds that were presented above.

The algorithm also has a few classes of weak keys. Detection of a key belonging to these classes requires only two chosen plaintexts encryptions. Additionally, IDEA is claimed to have many other weak key classes, given its present key-schedule mechanism. Such weak keys should be avoided in any implementation.

Furthermore, with regards to weak keys, 2^{23} keys exhibit a linear factor. A linear factor is a linear equation between the plaintext, the key and the ciphertext, that applies for all input. Membership in this group can be detected by observing some plaintext-ciphertext pairs. In addition to this class, 2^{35} keys have global characteristics[7]. Additionally, in another group of 2^{51} keys, these are easily recoverable if detected as belonging to this group. Such membership detection takes two encryptions and the solution of 16 linear equations. All these weak keys must be detected and avoided by the implementation.

There also exists a related-key ciphertext-only attack on IDEA that requires $5*2^{17}$ related-keys encryptions, each on 2^{20} random (unknown) plaintexts.

ASYMMETRIC CIPHERS

The following sub-chapter presents the weaknesses that were found in algorithms for a*symmetric encryption*. An asymmetric encryption algorithm is an algorithm by which decryption and encryption are performed using two separate (but mathematically related) keys. Such algorithms are often called "*Public key cryptosystems*".

[6] One half round in IDEA represents the exit permutation.

[7] For such keys, a known relationship between inputs is mapped to a known relationship between the outputs

RSA

RSA (named after Rivest, Shamir and Adelman) is the most commonly used asymmetric encryption algorithm, being adopted by many products since the 1970's. Its strength relies on the mathematical complexity of prime-factoring, which is still high enough to offer robustness when large primes are used. No severe inherent vulnerabilities have been found in RSA and all vulnerabilities that have ever been presented refer to a specific implementation of the algorithm. Although the algorithm is safe by its academic nature, it is probably the most susceptible to weaknesses due to weak implementation. These restrictions on implementations of the algorithm are hereby presented, under the assumption that the reader is familiar with the RSA basics and notations.

RSA private keys (hereafter called "*Private Exponents*") are likely to be weak if their value is less than $N^{0.292}$. This figure is occasionally adjusted and it is believed that practical secure implementations require the private exponent to be larger than $N^{0.5}$. This latter figure has not been proven.

The system *(N,d,e)* is likely to be insecure if *(p-1)*, for the *p* that is one of the factors of *N*, is a product of small primes.

When implementing an RSA system with several key-pairs, the implementer often chooses to use the same *N* for all key-pairs, thus saving computation time. However, since the private and public exponents together always assist in factoring *N*, every single member of the system will be able to factor *N* with his key-pair and use the result to invert any public exponent to the corresponding private exponent. Therefore, it is necessary to generate a new *N* value for each key-pair.

When using RSA for signature, the *blinding* feature[8] of the algorithm may be used by an attacker to cause a person to sign a message that he/she would not willfully sign. Applying a hash function on the message prior to signing can effectively solve this problem. This will allow the blinding feature but will make the blinding-based attack infeasible.

Whereas it is obvious that the system is weak when the private exponent is small, it has also been proven that the system is weak if the public exponent is too small. This claim is backed up by the proven *Coppersmith Theorem* and the *Hastad Broadcast attack*. Low public exponents are also risky when related messages are encrypted with the same public key. Such related messages are messages for which there is a polynomial that converts one message to the other. Furthermore, the related-messages attack was expanded so that it applies to two messages with different random padding (even when the padding values are unknown), as presented by Coppersmith in the *Short-pad attack*.

It is important to keep the entire private exponent (private key) secure as a whole. There are attacks that enable an adversary to recover the remaining bits of a private key using known bits. If the public exponent is smaller than $N^{0.5}$, then the private key can be recovered from a fraction of its bits.

As with many other algorithms, RSA is vulnerable to timing attacks that are easily implemented, as well as to power consumption attacks. The "Repeated Squaring Algorithm" can be used to effectively mount a timing attack. The solutions against timing attacks are either to add artificial delays or to use blinding.

When RSA is implemented using the Chinese Remainder Theorem (CRT) to save computation time, a breach can be formed if the signature algorithm fails for

[8] A description of the blinding feature is beyond the scope of this document. It will only be noted that this feature enables an entity to sign a message without knowing its content.

some reason. In this case, the attacker may be able to factor *N* using the mistaken signature that is generated. This problem can be solved either by adding random padding to the message, so the attacker never knows the exact message that was signed, or by verifying the signature prior to presenting it, in order to ensure that it was calculated properly.

When RSA is implemented according to PKCS#1, a standard block represents an RSA encrypted message. This block starts with a fixed constant ("02" in this case) that symbolizes an encrypted block. When a server receives a message, it decrypts it and issues an error message if these first two bytes don't match this fixed value. According to *Bleichenbacher's oracle* attack, this "hint" in the form of an error message is enough for an attacker to mount some sort of a brute-force attack. It is therefore important to implement a system that does not generate public error messages when a fixed block is not encountered.

If *p* and *q* that are used to generate *N* are too close to each other, then *Fermat's factoring* is possible, making the system highly insecure. Thus, the difference between the two primes should be at least $N^{0.25}$.

Diffie-Hellman (DH)

Diffie-Hellman is the one of the most common asymmetric algorithms. It is mainly used for two anonymous parties, who do not have a secure channel between them, to efficiently and securely exchange symmetric keys that will be used for the symmetric encryption of session data.

Diffie-Hellman is considered to be secure. However, its nature of being an algorithm which retains anonymity, implies that it is highly vulnerable to *man-in-the-middle* attacks. Diffie-Hellman, when not combined with adequate authentication methods, may therefore be highly risky due to these kinds of attacks.

HASH FUNCTIONS

Hash functions are functions that receive an arbitrary-length input and produce a fixed-length output. These functions are usually used for authentication and for digital signatures. Cryptographic hash functions are required to be *collision free*[9] and *non-invertible*[10].

MD5 (Message Digest 5)

MD5 is one of the most commonly used message digest (hash) functions. The MD5 algorithm is also known to be secure. No one has yet presented an efficient method for *inverting* the function. The function is also known to be *collision-free.*

[9] A hash function is regarded as "collision-free" if one cannot find two messages for which the function generates the same digest.

[10] Inverting a hash function refers to the action of recovering parts of the original message (input) from the digest (output). Complete inversion is, of course, impossible by the definition of a hash function as a function from a group of size aleph-zero to a group of a finite size.

The closest attack found was the ability to generate a pseudo-collision[11] in 2^{16} operations. However, this does not seem to cause any notable risk.

SUMMARY

Whereas most of the algorithms do not suffer from real vulnerabilities that make their implementation useless, some suffer from some kind of vulnerability that may put the user at risk in some circumstances. Still, it must be remembered that cryptographic forums are mostly academic, so they often publicize weaknesses that have purely academic value. Most of the weaknesses that were detected in algorithms that are known to be secure, have a high educational and professional value, but usually cannot be exploited in practice. Yet, such academic evaluation is perhaps the most reliable source for information about an algorithm's strength. Therefore, algorithms for which no weaknesses were published are not necessarily secure algorithms but most likely are algorithms that were never examined by professional entities.

In our view, the algorithms can be divided into three groups with respect to their strength, in ascending order as follows.

1. Algorithms that were examined by the academia (or by respectable cryptographers) and in which serious exploitable flaws were found (such as the CMEA). Also included in this group are algorithms that were not examined by the academia for reason of being too weak to start with. Algorithms belonging to this group are the Cellular Message Encryption Algorithm, Single DES, RC2 and RC4. Implementation of algorithms of this group is highly discouraged.
2. Algorithms that were not examined by the academia (or by any other group of respectable cryptographers) either for their lacking interest to the public or for their not being open-sourced. Algorithms belonging to this group are (by the definition of this group) not present in this document. Ciphers belonging to this group should be implemented only if there is a need for these specific ciphers, or if there is a basis for the assumption that the lack of publications about the strength of the algorithm is temporary.
3. Algorithms that were examined by the academia (or by any other group of respectable cryptographers) and for which no weaknesses were found (an uncommon situation) or for which the only weaknesses that were detected are not exploitable. Weaknesses are usually not exploitable if they require infeasible work time, amount of required known plaintext which exceeds the number of possible plaintext blocks, enormous amounts of related-key searches or chosen plaintexts, etc. Another category of non-exploitable weaknesses is weaknesses in reduced-round variants, in cases where it is clear that the weakness cannot be extended to the full-version of the cipher. Clearly, algorithms of this group are recommended for use in practice.

[11] A pseudo-collision is a case in which an attacker can predict two keys that will generate the same hash value for a single given message, when using keyed hash.

BIBLIOGRAPHY

- Dan Boneh, *Twenty Years of Attack on RSA*
- Dan Boneh and Glenn Durfee, *Cryptanalysis of Low-Exponent RSA*
- Fauzan Mirza, *Linear and S-Box Pair Cryptanalysis on DES*
- Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*
- Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson, *On the Twofish Key Schedule*
- Lars R. Knudsen, Vincent Rijmen, Ronald L. Rivest, and M.J.B. Robshaw, *Cryptanalysis of RC2*
- Lars R. Knudsen and Willi Meier, *Correlations in RC6*
- Sean Murphy and Matt Robshaw, *New Observations on Rijndael*
- John Kelsey, Bruce Schneier, and David Wagner, *Key Schedule Cryptanalysis*
- Henry Gilbert and Marine Minier, *A Collision Attack on Seven Rounds of Rijndael*
- John Kelsey, Bruce Schneier, and David Wagner, *Related-Key Cryptanalysis*
- Eli Biham and Adi Shamir, *Differential Fault Analysis on Secret-Key Cryptosystems*
- Eli Biham and Nathan Keller, *Cryptanalysis on Reduced Rijndael*
- Benne De Weger, *Cryptanalysis of RSA with Small Prime Difference*
- David Wagner, Bruce Schneier, and John Kelsey, *Cryptanalysis of the Cellular Message Encryption Algorithm*
- Doug Whiting, John Kelsey, Bruce Schneier, David Wagner, Niels Ferguson, and Chris Hall Further, *Observations on the Key-Schedule of Twofish*
- Eli Biham and Adi Shamir, *Differential Cryptanalysis of Full DES*
- Eli Biham, Alex Biryukov, *An Improvement of Davies Attack on DES*
- H.M. Heys and S.E. Tavares, *On the Security of the CAST Encryption Algorithm*
- C. Adams, H.M. Heys, S.E. Tavares, M. Wiener, *An Analysis of the CAST-256 Cipher*
- Peter C. Weiner, *Using Content-Addressable Search Engines to Break DES*
- Joan Daemen, Rene Govaerts, and Joos Vandewalle, *Cryptanalysis of 2.5 Rounds of IDEA*
- Bruce Schneier, *MD5 Cryptanalysis*
- Joan Daemen, Rene Govaerts, and Joos Vandewalle, *Weak Keys of IDEA*
- Johan Borst, Lars R. Knudsen, Vincent Rijmen, *Two Attacks on Reduced IDEA*

About Discretix

Discretix is a semiconductor intellectual property company that develops and licenses advanced embedded security solutions for resource-constrained environments, such as wireless devices and smart-cards, where stringent limits apply to the cost, size and power consumption of the target devices.

Discretix technology has already been adopted by some of the major vendors of wireless baseband and application chipsets.

Discretix Technologies Ltd.

Corporate Headquarters
43 Hamelacha Street
Beit Etgarim
Poleg Industrial Zone
Netanya 42504
Israel
Tel: +972 9 885 8810
Fax: +972 9 885 8820
Email:
marketing@discretix.com

Representative in Japan:
Triangle Technologies KK
Sogo-Hirakawacho Bldg.
4F 1-4-12
Hirakawacho Chiyoda-ku
Tokyo, Japan
Te l: +81 3 5215 8760
Fax: +81 3 5215 8765
Email:
japan.info@discretix.com