Probability of collisions in Soft Input Decryption

Nataša Živić, Christoph Ruland

***Abstract*—** In this work, probability of collision in Soft Input Decryption has been analyzed and calculated. Collisions of cryptographic check values cause wrong verification results. Therefore, it is important to find an analytical form of a probability of collisions, which can be used for estimation of an efficiency of Soft Input Decryption. It is known, that shorter cryptographic check values cause often collisions. For that reason, the number of collisions for cryptographic check values has been tested and compared with the theoretical results.

***Keywords*—**Collisions, Probability of Collisions, Probability of a Match, Soft Input Decryption.

I. Introduction

CHANNEL coding is a constitutional part of communication systems, which uses redundant information for the recognition or correction of errors that occur during the data transfer over a noisy channel. Cryptography is increasingly used in modern communication systems to provide secure information transfer, i.e. to protect against eavesdropping or manipulation of transmitted information, or masquerading of data origin.

The cooperation between channel coding and cryptography has been researched in [1], [2] and [3]: using channel decoding for the improvement of decryption results and, vice versa, using cryptography for the improvement of channel decoding. This concept is called Joint Channel Coding and Cryptography.

A message with a cryptographic check value is transmitted over a noisy channel using channel coding and decoding. The decryption of the cryptographic check value is very fragile, because all bits of the message and the cryptographic check value have to be correct. In case that one bit or more of the input of decryption is wrong, about 50 % of decrypted bits are false, and the verification of cryptographic check value fails. This problem can be solved using the method of correction which is studied in this work and called Soft Input Decryption: if the decoder is not able to reconstruct the original message and cryptographic check value because of a noisy channel or inefficiency of the channel decoding algorithm, it is possible to correct the message with the cryptographic check value using side information of the channel decoder in form of so called *L*-values.

Manuscript received February 9, 2007; Revised received July 11, 2007.

Nataša Živić is with the Institute for Data Communications Systems, Electrical Engineering and Informatics Department, University of Siegen, Siegen, 57076, Germany (e-mail: natasa.zivic@uni-siegen.de).

Christoph Ruland is the manager of the Institute for Data Communications Systems and a professor at the Electrical Engineering and Informatics Department, University of Siegen, Siegen, 57076, Germany (e-mail: christoph.ruland@uni-siegen.de).

Channel decoding can be improved using a message with cryptographic check value which has been corrected by Soft Input Decryption. This method is studied in this work as well and it uses corrected *L*-values as feedback information to the channel decoder for improved decoding of those bits which have not been yet corrected. The feedback method is iterative, because *L*-values corrected in one round are used for the correction of bits in the next iteration.

Collisions of cryptographic check values cause wrong verification results. The probability of collisions has been calculated in this paper.

II. Cryptographic Mechanisms of Data Integrity and Data Origin Authentication

Data integrity is the property that data have not been altered or destroyed in an unauthorized manner [4, 5]. As data can be changed during the transfer or storing phase, it is important to check that no modification happened until they were received.

Data origin authentication is the corroboration that the source of data received is as claimed [5, 6]. It is the cryptographic service, which proves the identity of the data origin, i.e. that data were indeed sent by the entity which is assumed to be the originator.

Hash values, MAC/H-MACs and digital signatures are considered as redundancy values in this work, because they have different lengths which influence the coding gain, code rate and probability of collisions.

A. Hash Functions

A hash function is a one-way function which maps strings of bits of variable length to fix-length strings of bits, satisfying two following properties:

- for a given output, it is computationally infeasible to find an input which maps to this output and
- for a given input, it is computationally infeasible to find a second input which maps to the same output [7].

The same standard defines a hash code as the string of bits which is the output of the hash function.

A collision resistant hash function is defined as a hash function satisfying the following property: it is computationally infeasible to find any two distinct inputs which map to the same output. Computational feasibility depends on the specific security requirements and environment [7].

Collision resistant hash functions are used for the generation of digital signatures.

The most commonly used lengths of hash value are 160, 228 and 256 bits. In this case, the collision probability is greater than 0.5 after about 2^{80} randomly chosen input messages according to the birthday paradox.

B. Message Authentication Codes (MAC)

MAC is an application of a symmetric block cipher [4]. Examples of used block cipher algorithm are DES [8], 3–DES and AES [9].

ISO/IEC 9797-1 specifies MAC algorithms that use a secret key and an n-bit block cipher to calculate an *n*-bit MAC. These mechanisms can be used as data integrity mechanisms to verify the fact that data have not been altered. MAC provides only subjective authentication, because identity of data origin cannot be proven by a third party (at least two parties are able to generate the same MAC) [10].

MAC can only be used as a message authentication mechanism to provide assurance that a message has been originated by an entity in possession of the secret key.

A MAC algorithm is a function which maps a string D of bits and a secret key K to fixed-length strings of bits, satisfying the following properties [4]:

- for any key and any input string the function can be computed efficiently
- for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute a function value on any new input string.

It should be noted that the birthday paradox applies also on MACs.

Typical length of the MAC is the block length of the block cipher, i.e. 64 or 128 bits. Details about the MAC algorithm are given in [4].

C. Hashed –Message Authentication Codes (H-MAC)

[11] specifies MAC algorithms that use a secret key and a hash function (or its round - function) with an n-bit result to calculate an *m*-bit MAC. These mechanisms can be used as data integrity mechanisms to verify that data have not been altered in an unauthorized manner. They can also be used as message authentication mechanisms to provide assurance that a message has been originated by an entity in possession of the secret key [11].

The length of H–MAC is the same as that of underlying hash function: 64, 160, 228 or 256 bits, but the length can be adjusted as necessary. For example, for hash functions RIPEMID–160 and SHA–1 the length of H–MAC is 160 bits.

Collision resistance of H–MAC is defined as for hash function (see chap. II *A.*).

An H-MAC algorithm (or hashed cryptographic check function) computes a function which maps string *D* of bits and a secret key *K* to fixed-length strings of bits (H-MAC or hashed cryptographic check value), satisfying the following properties [11]:

- for any key and any input string the function can be computed efficiently
- for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute a function value on any new input string.

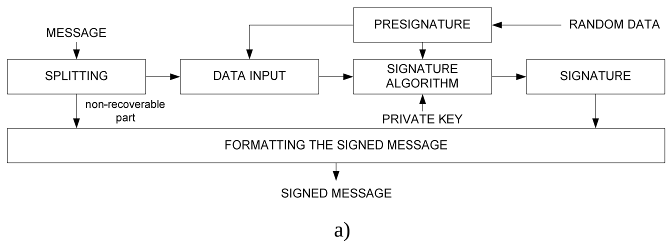
Details about the H-MAC algorithm are given in [11].

D. Digital Signatures

Digital signatures provide data origin authentication and support non-repudiation services. They normally use asymmetric cryptography, even if there are solutions for symmetric algorithms based digital signatures [12].

There are two types of digital signatures:

1. signatures giving message recovery (Fig. 1) [13]
2. signatures with appendix (Fig. 2) [14, 15].



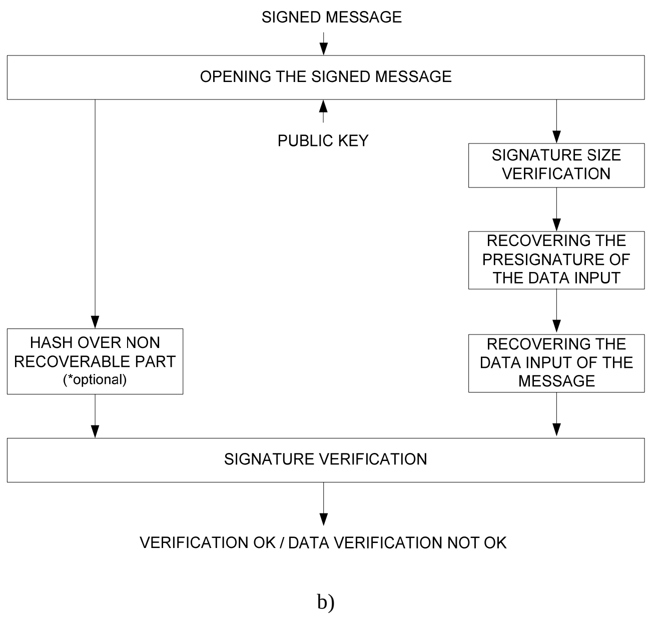
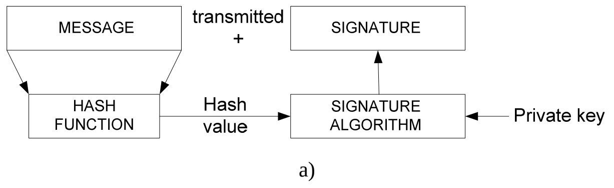

SIGNED MESSAGE
OPENING THE SIGNED MESSAGE
PUBLIC KEY
SIGNATURE SIZE VERIFICATION
RECOVERING THE PRESIGNATURE OF THE DATA INPUT
HASH OVER NON RECOVERABLE PART (*optional)
RECOVERING THE DATA INPUT OF THE MESSAGE
SIGNATURE VERIFICATION
VERIFICATION OK / DATA VERIFICATION NOT OK
b)

Fig. 1 Digital Signatures giving message recovery (simplified): a) Generation b) Verification

Signatures giving message recovery can be applied to "short" messages, which are extended by a onetime pre-signature before the execution of the signature operation (see Fig. 1). The decryptor recovers the message from the signature, if the signature is proved to be correct. "Short message" means, that the length of the message plus redundancy is shorter than the length of the private key used in the signature algorithm. If the message does not contain enough redundancy for verification, it is added by use of a hash function. If the message is too long, then message recovery is partial. In this case the message is divided into recoverable part (included in the signature) and non-recoverable part (stored and/or transmitted along with the signature) [13].

In the case of digital signatures with appendix, the message has an arbitrary length (Fig. 2). The encryptor generates a digital signature over a hash value which has been calculated over the message to be signed. The decryptor computes the hash value over the received message and verifies the signature by using the public key. The result of the signature verification is true or false.



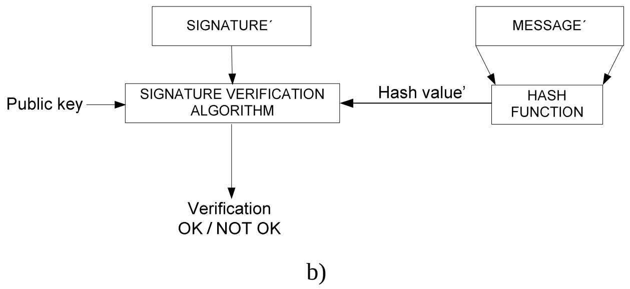


Fig. 2 Digital Signatures with appendix (simplified):

a) Generation b) Verification

In both cases, the verification result is negative if the input of the signature verification compared to the output of the signer is modified, or the public key and private key do not belong to the same key system. Digital signatures and messages - as input to the decryptor - have to be delivered from the channel decoder free of errors or modifications to verify the signature successfully.

III. Soft Input Decryption

The basic technique which is described and used in this work is called Soft Input Decryption. It consists of a decryptor which uses soft output of the channel decoder as soft input. [1].

The cryptographic mechanism which is used by encryptor and decryptor generates and verifies cryptographic check values (hash values, digital signatures, MACs, H-MACs) providing data integrity, data origin authentication and non repudiation.

The algorithm of Soft Input Decryption (Fig. 3) is as follows:

The decryption is successfully completed, if the verification of the cryptographic check value is successful, i.e. the output is “true”. If the verification is negative, the soft output of the channel decoder is analyzed and the bits with the lowest $|L|$-values are flipped (XOR “1”), then the decryptor performs the verification process and proves the result of the verification again. If the verification is again negative, bits with another combination of the lowest $|L|$-values are changed. This iterative process will stop when the verification is successful or the needed resources are consumed.

In case that the attempts for correction fail, the number of errors is too large as a result of a very noisy channel or an attack, so that the resources are not sufficient to try enough combinations of flipping bits of low $|L|$-values.

It may happen that the attempts for correction of SID block succeed, but the corrected cryptographic value is not equal to the original one: a collision happens. This case has an extremely low probability when cryptographic check values are chosen under security aspects. Collision aspects of cryptographic check values in Soft Input Decryption are the subject of this paper.

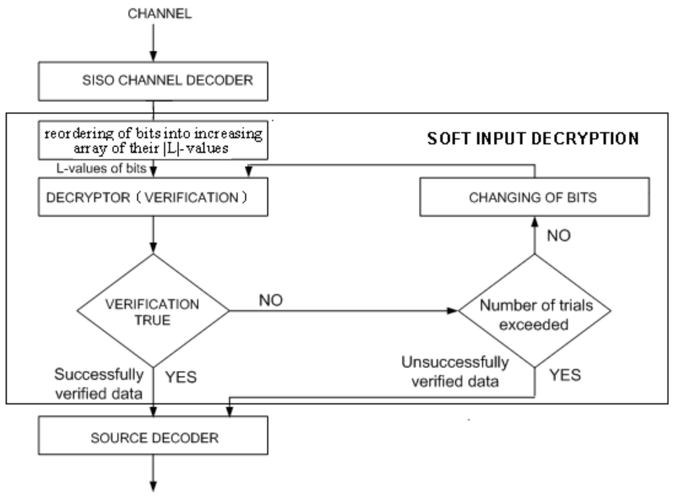


Fig. 3 Algorithm of the Soft Input Decryption

Soft Input Decryption is block oriented. The block which is taken from sequential input bits to the channel encoder and should be corrected by Soft Input Decryption after channel decoding is called SID block (Soft Input Decryption block). The SID block may have different contents depending on cryptographic mechanisms and scenarios [2].

IV. Collisions

Collisions caused by changes of bits of a message and of a redundancy check value by Soft Input Decryption (SID) will be analyzed and calculated in this paper.

A redundancy check value *RCV* is a cryptographic check value *CCV* (digital signature, MAC, H-MAC), if cryptographic check functions are used or any other systematic redundancy value added to a message, for example hash value or CRC. The redundancy check value is the result of a redundancy check function *RCF*.

Note: A hash value has characteristics of a cryptographic check function, but the properties are of a redundancy check function if the hash value is not combined with any other cryptographic mechanism. A hash value itself provides no security.

The problem of collisions is described in Fig. 4.

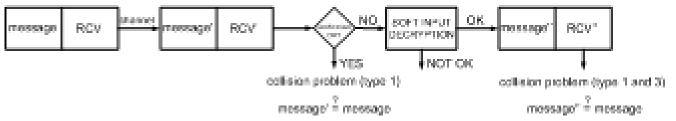


Fig. 4 Description of the problem

There are following types of collisions:

Type 1: It may happen that the first verification before Soft Input Decryption started is successful, although the originally sent message and the message which resulted in the successful verification are different. The condition for collision in these cases is:

$$[(message \neq message') \wedge (RCF(message') = RCV')] = true \tag{1}$$

The collision is caused by modifications during the transmission over the channel which could not be corrected by the channel decoder. This type of collision is not an implication of Soft Input Decryption. For this reason this collision will not be the subject of this paper.

Type 2: It may happen that *RCV"* is equal to the originally sent *RCV*, and verification is successful, although the originally sent *message* and *message"* are different. The condition for this type of collision is:

$$[(message \neq message'') \wedge (RCV = RCV'') \wedge (RCF(message'') = RCV'')] = true \tag{2}$$

This type of collision happens by changing bits by Soft Input Decryption and is the subject of this chapter. This type of collision represents a collision in the cryptographic sense defined for hash functions.

Type 3: It may also happen that *RCV"* differs from the originally sent *RCV*, and a *message"* differs from originally sent *message*, but the verification is successful. The condition for this type of collision is:

$$[(message \neq message'') \wedge (RCV \neq RCV'') \wedge (RCF(message'') = RCV'')] = true \tag{3}$$

Note: This type of collision does not represent a collision in the cryptographic sense defined for hash functions.

This condition causes wrong results, although the verification is successful. Therefore, this event is regarded as a collision. This type of collision happens by changing bits by Soft Input Decryption and is also the subject of this paper.

V. The Probability of Match

A match is the event, that the verification is successful:

$$RCF(message'') = RCV'' \tag{4}$$

The probability of a match is defined as:

$$P_{match} = P[RCF(message'') = RCV''] \tag{5}$$

in any trial of Soft Input Decryption.

The correct match $P_{correct}$ is the case that the message and redundancy check value are corrected by Soft Input Decryption and that no collision happened:

$$[(message = message'') \wedge (RCF(message'') = RCV'')] = true \tag{6}$$

The probability of a match between the message and redundancy check value will be calculated under following assumptions:

1. all redundancy check values have the same probability of appearance
2. bits which are changed in Soft Input Decryption are randomly distributed over the message and redundancy check value.

*Note: This assumption is true for CRC, but it is not proven that it is true for, for example, hash values.

p(match, i) is the probability that a match happens when *i* bits of *j* changed bits are in the message part. Because of assumption 2., the probability that *i* bits of *j* flipped bits are located in the message part of length of *m* is given by hypergeometric distribution [10]:

$$p_m(i) = \frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}}, i = 0,1,..., j \tag{7}$$

For example, if *j* = 16 and *m* = *n* (Fig.53), distribution of *i* is symmetrical around the most likely value *i* = 8, which means that it is most likely that the same number of changed bits is located in the message and in the redundancy check value, as expected. For *m* > *n* the distribution is not symmetrical and positions of lowest |*L*| -values are mostly in the message. In the case presented in Fig. 4 with *m = 2n*, it is most likely that 11 positions of the lowest $|L|$ – values are placed in a message, i.e. 5 positions are in redundancy check value.

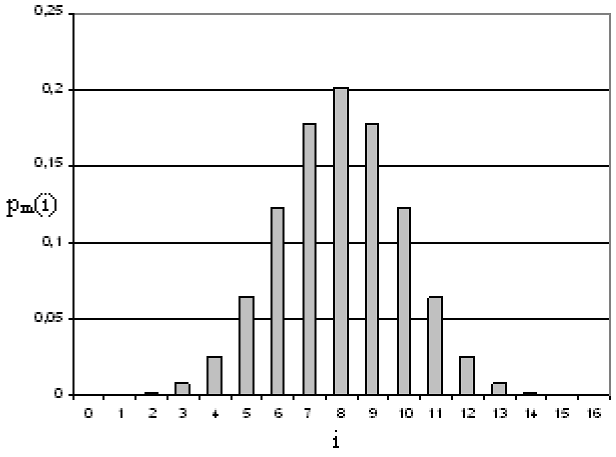


Fig. 5 Probability $p_n(i)$ when $m = n$ ($j = 16$)

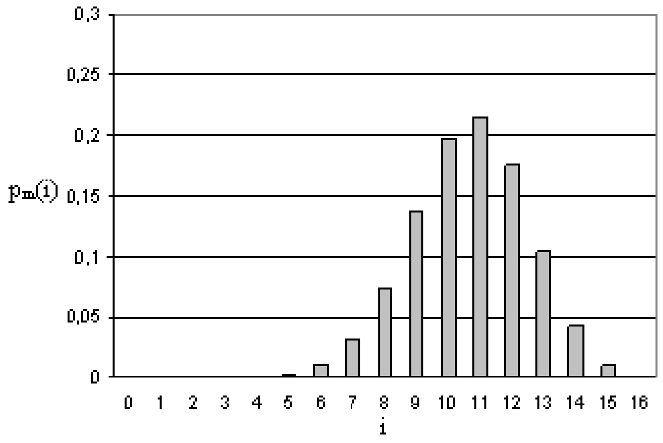


Fig. 6 Probability $p_n(i)$ when $m = 2n$ ($j = 16$)

The probability of a match when i bits with the lowest $|L|$-values are in a message part is $p(match|i)$. The probability of a match $p_{match,j}$ when j bits are flipped, can be calculated as the sum of probabilities of matches for each i:

$$p_{match,j} = \sum_{i=0}^{j} p(match,i) = \sum_{i=0}^{j} p_m(i) p(match|i) \quad (8)$$

Flipping j chosen bits within the Soft Input Decryption algorithm, a set of 2^i different messages and a set of 2^{j-i} different redundancy check values are obtained. The redundancy check function of each of 2^i messages might be equal to any of 2^{j-i} produced redundancy check values. The probability that any of 2^{j-i} redundancy check values match to one specific message is:

$$p_{i,match} = \frac{2^{j-i}}{2^n} \quad (9)$$

and the opposite probability that no match with a specific message occurs is:

$$\overline{p}_{i,match} = 1 - p_{i,match} \quad (10)$$

After 2^j attempts of tests, with i bits in the message part, the probability that no match occurs is:

$$\overline{p(match\,|\,i)} = (\overline{p}_{i,match})^{2^j} = \left(1 - \frac{2^{j-i}}{2^n}\right)^{2^j} \quad (11)$$

So, the probability of a match after 2^j attempts with i bits in the message part is:

$$p(match\,|\,i) = 1 - \left(1 - \frac{2^{j-i}}{2^n}\right)^{2^j} \quad (12)$$

When the number of flipped bits is increased from j-1 to j, 2^{j-1} tests are performed, because these tests have not been performed before. If the additionally flipped bit is in a message part, it happens with the probability of $\frac{m-i}{m+n-j}$. Vice versa, if the additionally flipped bit is in a redundancy check value part, it happens with the probability of $\frac{n-(j-i)}{m+n-j}$. Finally, the probability of a match after flipping up to N bits is given as:

$$P_{match} = A + B, \quad (13)$$

where

$$A = \sum_{j=1}^{N} \sum_{i=1}^{j-1} \frac{n-(j-i)}{m+n-j} \frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}} \left[1 - \left(1 - \frac{2^{j-i}}{2^n}\right)^{2^{i-1}}\right] \quad (14)$$

and

$$B = \sum_{j=1}^{N} \sum_{i=1}^{j-1} \frac{m-i}{m+n-j} \frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}} \left[1 - \left(1 - \frac{2^{j-i}}{2^n}\right)^{2^{i-1}}\right] \quad (15)$$

VI. The Probability of Collisions

A collision as an implication of Soft Input Decryption happens in case 2. (collision type 2) and 3. (collision type 3) of Chapter III and can be calculated as:

$$P_{coll} = P_{match} - P_{correct} \tag{16}$$

The correct match ($P_{correct}$) is one of 2^{m+n} possible matches of messages and redundancy check values:

$$P_{correct} = \frac{1}{2^{m+n}} \tag{17}$$

The collision probability is, using equations (11), (12), (15) and (16):

$$P_{coll} = A + B - P_{correct} \tag{18}$$

where A and B are given by equations (14) and (15), respectively.

VII. Collision Tests

Collision tests were performed by simulations of Soft Input Decryption using short redundancy check values (up to 24 bits) and messages of various lengths, for N = 8. All simulations are programmed in C/C++ programming language. For each point of the curves 50 000 tests are performed, which is enough for reliability of results [16]. The transfer of the SID block is simulated by the use of an AWGN channel. The used convolutional encoder has a code rate r = 1/2 and constraint length m = 2 (Fig. 7). The decoder uses a MAP algorithm [17].

SHA-1 hash function (160 bits) is used as a redundancy check function. Shorter redundancy check values used for tests are got by taking right most bits of the hash value. Soft Input Decryption tests stopped after the first successful verification. After each verification it is checked if the verification is correct or a collision happened. So, the number of collisions is counted.

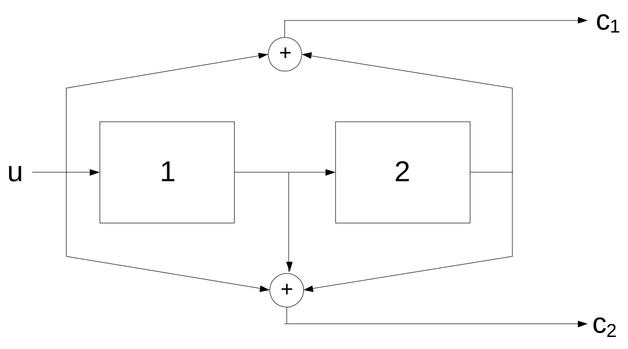


Fig. 7 Convolutional encoder *r = ½, m = 2*

The results of tests are shown in Fig. 8 in comparison to the results of equation (18).

The results of equation (18) depend mainly on the length of the redundancy check value, and they stay almost constant, if the length of the message changes (the results change on 7. or higher decimal position). Tested collision probability of Soft Input Decryption depends also on the length of the redundancy check value and has no significant change (on 4. or higher decimal place) with the change of message length, but it is lower than the collision probability calculated by equation (18). The reason that tested collision probability is lower than the one of equation (18) is that the equation (18) is got assuming random combinations of changed bits. Soft Input Decryption uses |L|-values to find the correct message and not random combinations of changed bits, so that the probability of the correct match is much higher than that in equation (17).

For that reason, the results of equation (18) – theoretic results in Fig. 8, can be used as “the worst case”, i.e. an upper limit of the collision probability.

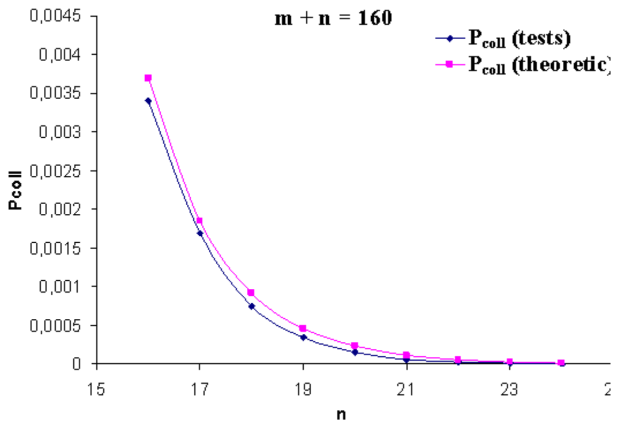


Fig. 8 Comparison of collision results of tests and theoretical results for up to 2^8 trials

VIII. Conclusion

This paper analyzes probability of collisions which can happen using Soft Input Decryption. Collisions are standard problem in cryptography, as they implicate wrong verification results. Probability of collisions grows as the length of cryptographic check values decrease.

Computation of probability of collision has been performed by subtraction of the probability of match and probability of the correct match. Additionally, simulations have been performed for comparison of analytical (theoretical) results and results of tests. The comparison shows that analytical results can be used for estimation of the efficiency of Soft Input Decryption, as the upper bound of probability of collisions of Soft Input Decryption.

References

[1] N. Živić, C. Ruland, “Soft Input Decryption”, *4th Turbocode Conference, 6th Source and Channel Code Conference*, VDE/IEEE, in Plastics, Munich, April 2006.

[2] N. Živić, C. Ruland: “Channel Coding as a Cryptography Enhancer”, *Advances in Communications, Proceeding in the 11th WSEAS international conference on Communications (part of the 2007 CSCC multiconference)*, Agios Nikolaos, Crete Island, Greece, July 2007.

[3] N. Živić, C. Ruland, “Feedback in Joint Coding and Cryptography”, *7th International ITG Conference on Source and Channel Coding VDE/IEEE*, Ulm, January 2008.

[4] ISO/IEC 9797-1, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*, 1999.

[5] ISO/IEC 13888-1*, Information technology – Security techniques – Non-repudiation – Part 1: General*, 2004.

[6] ISO/IEC 9798-1, *Information technology – Security techniques – Entity authentication mechanisms – Part 1: General*, 1997.

[7] ISO/IEC 10118-1, *Information technology – Security techniques – Hash-functions – Part 1: General*, 2000.

[8] ISO/IEC 8372, *Modes of operation for 64-bit block cipher algorithm*, 1987.

[9] ISO/IEC 18033-3, *Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers*, 2005.

[10] C. Ruland, *Informationssicherheit in Datennetzen*, Datacom Verlag, Bergheim, 1993.

[11] ISO/IEC 9797-2, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a hash- function*, 2000.

[12] C. Ruland, “Realizing digital signatures with one-way hash function”, *Cryptologia*, *Vol XVII, Number 3*, July 1993.

[13] ISO/IEC 9796-2, *Information technology – Security techniques – Digital signatures giving message recovery – Part 2: Discrete logarithm based mechanisms*, 2006.

[14] ISO/IEC 14888-1, *Information technology – Security techniques – Digital signatures with appendix – Part 1: General*, 1998.

[15] ISO/IEC 15946-4, *Information technology – Security techniques – Cryptographic techniques based on Elliptic Curves – Part 4: Digital signatures giving message recovery*, 2004.

[16] M. Jeruchim, P. Balaban, K. S. Shanmugan, “Simulation of Communication Systems”, *Kluwer Academic/Plenum Publ*, New York, 2000.

[17] L. Bahl, J. Jelinek, J., Raviv, F. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate”, *IEEE Transactions on Information Theory*, IT-20, March 1974.