Channel Coding as a Cryptography Enhancer

NATASA ZIVIC and CHRISTOPH RULAND
Institute for Data Communications Systems
University of Siegen
Hölderlinstraße 3, Gebäude E, D-57076 Siegen
GERMANY
natasa.zivic@uni-siegen.de http://www.dcs.uni-siegen.de/staff/zivic/index.php
christoph.ruland@uni-siegen.de http://www.dcs.uni-siegen.de

Abstract: - In this work, channel decoding is considered as a promising way for improvement of cryptographic functions. Use of MAC/H-MAC values and digital signatures is analyzed in a context of code concatenation, together with convolutional codes using MAP decoding algorithm. Soft Input Decryption method, which uses L-values from channel decoder, presents an efficient method for integrating cryptography into decoding. The results of computer simulations that implement this method have been also presented. Additionally, the number of L-values verifications has been tested and compared with the theoretical results.

Key-Words: - Soft Input Decryption, Code Concatenation, L-values, MAC/H-MAC, MAP, Channel Decoding, Turbo Decoding, CCER

1 Cryptography as the Part of Decoding Process

Encryptor and decryptor are nowadays standard components of communications systems and they are placed between the source and channel encoder at transmitter's side, and between channel and source decoder at receiver's side. They are used in cryptographic operations to support communication security i.e. data integrity and authentication of data origin.

Cryptographic techniques can also be used in combination with channel decoding in order to achieve better decryption results by correction of cryptographic check values.

The soft outputs (L-values as a measure of reliability of decoded bits) of SISO (Soft Input Soft Output) channel decoding can be used to correct cryptographic check values [1].

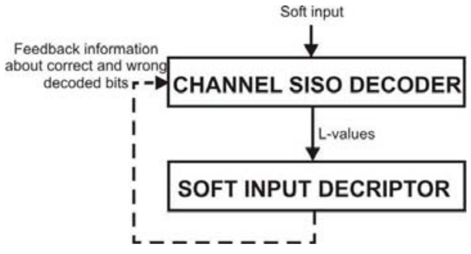

Fig.1 Concept of Soft Input Decryption

Soft output values provide very useful information about decoded bits, which find its place in today's fastest decoders: turbo decoders [2][3].

In contrast to turbo decoders which use feedback within channel decoder to get so called extrinsic information, in this work L-values from channel decoder have been used as information to the new entity – decrypting mechanism. In this way, cryptographic mechanism beside data integrity and authentication, also enables error correction (Soft Input Decryption method), and the redundancy from a usual encryption-decryption process is used now as an error correction code as well.

Digital signatures, MAC/H-MAC ((Hash)-Message Authentication Code) and hash values have been used as cryptographic check values. In the light of coding theory, those check values can also be considered as a code which is concatenated to the channel code.

Convolutional codes can be concatenated to RS (Reed-Solomon) or other convolutional code. Concatenated code can be SISO decoded in order to improve error performance of decoding system [4]. Such a type of concatenated codes can be compared to the combination of codes investigated in this work, as the code is convolutional and decoding is in a SISO convolutional decoding. Thus, two good characteristics are the result of such concatenated schema: good error performance because of the use of SISO principle and good security performances as the result of the use of the cryptographic mechanisms.

message (m bits)	cryptographic check value (n bits redundancy)

Fig.2 Message with the cryptographic redundancy

Cryptographic check values have different lengths, depending on the used cryptographic mechanisms. Those lengths define the amount of added redundancy, influencing overall coding rate (Fig.3):

$$R_{overall} = R_1 R_2 = \frac{m}{m+n} R \qquad (1)$$

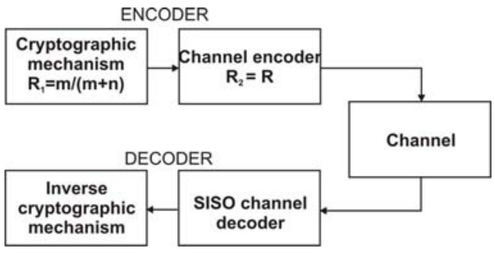
ENCODER
Cryptographic mechanism R₁=m/(m+n)
Channel encoder R₂= R
Channel
DECODER
Inverse cryptographic mechanism
SISO channel decoder

Fig.3 System overview

Coding rate is decreasing in this way, which is a price for added redundancy. As cryptographic check value has been added for security reasons, cryptographic redundancy is present anyway and in the same time exploited for error correcting.

2 Coding and Cryptography in Communication System

From the point of view of information theory, a usual communication system consists of a source, a source encoder, a channel encoder and a line encoder (for example modulator) on the sending side, the transmitting medium or the channel, and a line decoder (for example demodulator), a channel decoder, a source decoder and a sink on the receiving side.

Channel decoder decodes the line decoded data, by using the added redundancy. It performs hard decision (in the case of hard output line decoding) or soft decision decoding (in the case of soft output line decoding). The results of a soft decision decoder are better than the results of hard decision decoder. The output of the channel decoder can also be hard (the output bits are 1 or 0) or soft. SISO decoding is a concept of channel decoding, which was originally used in iterative and turbo coding, because the soft outputs were fed back internally. Soft output decoding is also used in joint source-channel decoding. Soft output is used in this work as soft input for the decryptor, which will be called Soft Input Decryptor. Soft outputs are usually expressed as L-values [5]:

$$L(u') = \ln \frac{P(u=+1)}{P(u=0)} \qquad (2)$$

The sign of L-value is a hard decision, and the |L|-value is used as reliability value of a hard decision. As higher the |L|-value, as the hard decision information is more reliable (L = ±∞ is the soft output for a correct decision) and vice versa: lower |L|-value means less reliable decision information. When the L-value is equal to 0, the probability of the correctness of the decision is 0.5.

Besides further sending of information from one to another entity, feedback of information is established with following benefits:

1. inside of the channel decoder the application of iterative decoding and Turbo codes is used for the improvement of error rates
2. the feedback information about the decoded bits from the channel decoder to the demodulator is used to improve the equalization and synchronization of the demodulator [3]
3. the feedback information from the source decoder to the channel decoder is used to improve the channel decoding (so called source channel decoding) [6].

When encryptor and decryptor are inserted (Fig.4), the feedback between source decoder and channel decoder is interrupted. On the other side, the concept of joint channel coding and cryptography provides such interfaces of encryptor/decryptor which enable their integration into the communication system without modification of source/channel encoders/decoders. Feedback from the source decoder can be used by the decryptor and the decryptor produces feedback to the channel decoder or forwards the feedback from the source decoder to the channel decoder.

Assuming soft output from the channel decoder, the decryptor is seen not only as a cryptographic module, but also as a decoding module that works transparently between channel and source decoder as the encryptor is transparently added between source and channel encoder (Fig.4) [1].

The decryptor can recognize or even correct errors if the encryptor adds redundancy. The errors are corrected when the decryptor successfully verifies cryptographic check values.

In the case that the decryptor cannot verify the cryptographic check values, it can forward the values received from the channel decoder to the source decoder together with the information that the verification was not successful.

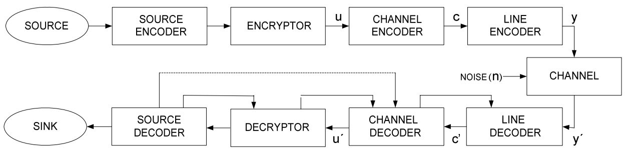

Fig.4 Modular block diagram of coding/decoding

Therefore, the decryptor is considered to work also as a part of overall decoding process. The security aspects and properties of encryption/decryption are also respected, if the cryptographic algorithm and context are chosen according to the security requirements.

2.1 SISO Channel Decoding

Preferred SISO decoding algorithms which are used for turbo decoding appeared much earlier than the turbo decoding itself: Maximum A-Posteriori (MAP) [7], which has been used in this work, and Soft Output Viterbi Algorithm (SOVA) [8]. The MAP algorithm gives better results than SOVA for low S/N ratio and both of them have similar decoding capabilities for high S/N. Nevertheless, MAP is much more complex than SOVA.

2.2 Cryptographic mechanisms of data integrity and data origin authentication

Data integrity is the property that data has not been altered or destroyed in an unauthorized manner [9]. 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 [10]. It is a 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.

3 Soft Input Decryption versus Cyclic Redundancy Check

Cryptographic check values will be used as Cyclic Redundancy Checks (CRCs) in the method of Soft Input Decryption.

CRC is a function used to produce a checksum of data longer than 1B which have to be transmitted. This checksum is used to detect errors after transmission. A CRC is computed and appended to data before transmission, and verified after transmission by the recipient to confirm that no changes occurred on transit.

message	CRC

Fig. 5 Message with CRC

Receiver recognizes several classes of errors:
- single error
- double errors
- all odd number of errors and
- all errors with the length smaller than degree of a generator polynomial.

CRC recognizes transmission errors, but does not correct them (in general). On the contrary, Soft Input Decryption is an error-correction method.

4 Definition of Soft Input Decryption Technology

Basic idea for the improvement of the decrypting mechanisms uses soft output of the channel decoder [1].

A decryptor which verifies cryptographic check values (digital signatures, MAC, H-MAC) has been chosen. Algorithm of Soft Input Decryption (Fig.6) is as follows:

The security mechanism is successfully completed on the receiving side if the cryptographic check value (digital signature, MAC, H-MAC) is recognized by the decryptor to be correct. If the verification is negative, the decryptor analyzes soft output of the channel decoder, changes the bits with the lowest absolute L-values, performs the verification process and checks the result of the verification again.

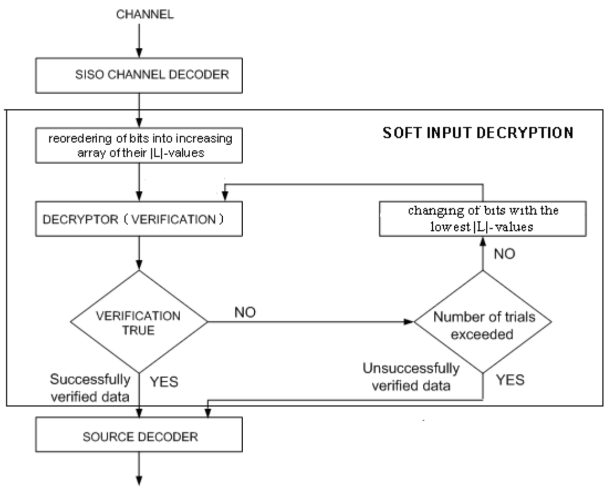
CHANNEL
SISO CHANNEL DECODER
reoredering of bits into increasing array of their |L|-values
SOFT INPUT DECRYPTION
DECRYPTOR (VERIFICATION)
changing of bits with the lowest |L|-values
NO
VERIFICATION TRUE
NO
Number of trials exceeded
Successfully verified data
YES
Unsuccessfully verified data
YES
SOURCE DECODER

Fig. 6 Algorithm of the Soft Input Decryption

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

It may happen that the attempts for correction of cryptographic check values succeed, but the corrected cryptographic value is not equal to the original one: a collision happened. This case has an extremely low probability when cryptographic check values are chosen under security aspects.

5 Strategy of Soft Input Decryption

The module "Changing of bits of SID block" (see Fig. 6) contains the strategy, in which sequence bits and combinations of bits of the SID block are changed, before the next verification is achieved. Depending on the strategy of Soft Input Decryption, different schedules of bit correction are possible.

The static strategy of Soft Input Decryption (chapter 5.1) is used for the result in this paper. Proposals for other strategies are given in chapters 5.2 and 5.3.

5.1 Static strategy

If the first verification after starting Soft Input Decryption is not successful, the bit with the lowest $|L|$-value of the SID block is flipped, assuming that the wrong bits are probably those with the lowest $|L|$-values. If the verification is again not successful, the bit with the second lowest $|L|$-value is changed. The next try will flip the bits with the lowest and second lowest $|L|$-value, then the bit with the third lowest $|L|$-value, etc. The process is limited by the number of bits with the lowest $|L|$-values, which should be tested. The strategy follows a representation of an increasing binary counter, whereby the lowest bit corresponds to the bit with the lowest $|L|$-value, etc.

The strategy is defined by following algorithm:

The static strategy orders the bits of the SID block by their $|L|$-values starting with the lowest one and monotony increasing. The output of the sort algorithm is represented as an example in Fig. 7.

j	1	2	3		w
P_j	78	35	112	.....	6

Fig. 7 Sorted sequence of bits of a SID block (example)

j is the increasing sequence of positions of $|L|$-values and P_j indicates the position of the bit with the j^{th} lowest $|L|$-value in the original SID block. The length of the SID block is w.

The function "Changing of bits of SID block" of the Soft Input Decryption algorithm (Fig. 3) is shown in Fig. 8 in detail. To control the strategy, an incrementing counter $i = 1, \ldots, 2^{Nmax} - 1$ is used in binary representation of fixed lengths of $N_{\max}$ with coefficients:

$$i = \sum_{j=1}^{N_{\max}} c_{i,j} 2^{j-1} \tag{3}$$

N_{max} is the maximum number of bits to be flipped, rsp. $2^{Nmax} - 1$ is the maximum number of trials, if all verifications fail. Each value of the counter i describes one trial. The indices j of those coefficients c_{ij} which are marked ($c_{ij} = 1$) indicate the positions of L-values of the bits to be flipped. These positions can be found by using the sorted table as in Fig. 7. The sorted sequence P_j can be limited to P_{Nmax}. i is reset to 0 at the beginning of Soft Input Decryption.

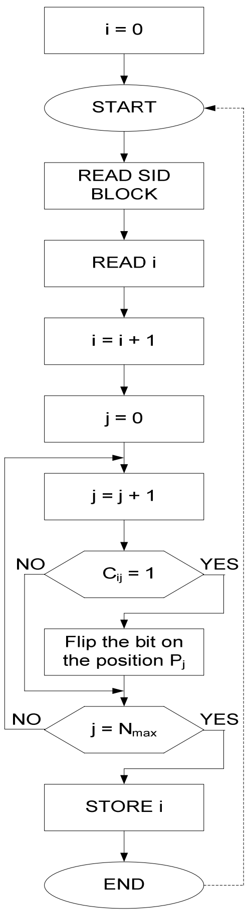

Fig.8 Algorithm of the static strategy

The strategy is based on the following assumption: if bits are wrong decoded by the channel decoder, than they have the lowest |L|-values. Unfortunately, this assumption is not true, because L-values are probability based and give only an orientation which bits may be wrong decoded. It may happen, for example, that combinations of up to 7 bits have to be tested, when only 3 bits are wrong.

5.2 Dynamic strategy

The static strategy sorts the L-values of single bits. The dynamic strategy calculates the L-values of groups of bits to decide which trial should be performed next. It can happen, that a group of bits result in a lower |L|-value than a |L|-value of a single bit, i.e. that a specific group of bits is probably wrong. By this way, it is possible to find the group of wrong bits in only few trials, not testing all combinations of flipped bits until the right combination is found like in the static strategy. The calculation is based on use of the L algebra [5] and much more complex than the static strategy. The elaboration of the dynamic strategy is for further study.

5.3 BER based strategy

The BER based strategy analyses which number of errors is the most probable in the SID block, under consideration of E_b/N_0 rsp. BER. To reduce the potential number of tests, L-values are taken into account. Example: if N_{max} is 16 and 4 bit errors are most probable for SID blocks of 320 bits, then up to $\binom{16}{4}$ instead of $\binom{320}{4}$ tests are performed. If $\binom{16}{4}$ tests are not successful, $\binom{16}{3}$ tests of 3 bit errors and $\binom{16}{5}$ tests of 5 bit errors have to be tested. The BER strategy is also for further study.

6 Applications of Soft Input Decryption

As examples, three possible applications of SID have been discussed. Let's assume **m** is the length of the message and **n** is the length of the cryptographic check value (redundancy).

6.1 Application 1

This application (Fig.9) uses digital signatures giving message recovery [11]. In this case, the signature contains the message plus redundancy. The length of the message is limited by the lengths of the redundancy and the signature input. If the signature is found to be correct after transmission, the recovered message is also correct. This scenario can be often found in transaction oriented applications exchanging short messages (in the case of no collision), which have to be authentic.

Typical examples are measurement values in industrial metering systems (electricity, water, gas etc.), as well as in stock exchange rates and other bank transactions.

Examples of signature algorithms which can be used are: ECDSA and ECNR.

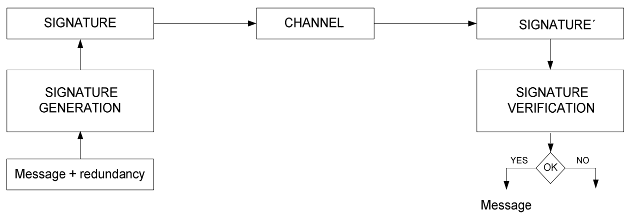

Fig. 9 Application with signatures giving message recovery

6.2 Application 2

Second application (Fig.10) considers signatures with appendix [12]. After a failed verification of a signature, it is not known whether the message or the signature was modified. In this scenario, it is assumed that the message and the signature have been transmitted separately: the message can be transmitted via different communication channel from the one used for digital signatures (outband), or via the same communication channel (inband). In the case of outband, the message itself is transmitted first and, if modified by the communication channel, corrected by redundancy within the message, by repetition or by agreement of communication partners. So, it is assumed that the message is transmitted correctly to the receiver. The signature is transmitted afterwards, either when it is requested, generated or when the action described by the message should be executed.

Typical examples are contracts or bank transactions, which are prepared in advance, and the digital signature is transmitted at the requested moment. In the case that an error occurs during the signature verification, the signature is not correct: it has been manipulated or errors occurred during the transmission, which could not be corrected by the channel decoder.

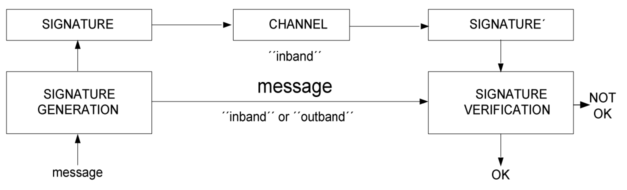

Fig. 10 Application with signatures with appendix

6.3 Application 3

This application (Fig.11) considers messages with a cryptographic check sum (MAC [13] or H-MAC [14]) or digital signatures with appendix.

Typical examples are credit/debit applications and other bank applications.

Application 3 is similar to application 1, because in both cases the correct message is delivered from the verification process. The difference is that the message can be read in a clear text in application 3, even if it was modified, while in first application the message cannot be recovered if there was a transmission error. On the other hand, block lengths used in third application are bigger than in application 1.

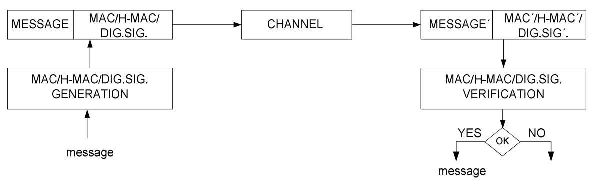

Fig. 11 Application with messages with a

7 Decrypting gain

Simulations can be performed for any of above described applications.

Soft Input Decryption verifies the relation between a message and a cryptographic check value.

Application 2 has been used for the implementation of simulations.

Tests use digital signature with appendix algorithm with SHA-1 as hash function and ECDSA (DSA on elliptic curves) over GF (p) with a length of p of 160 bit [15]. The length of the signature is 321 bit. The transfer of the signature is simulated by AWGN channel. The implemented convolutional encoder has r = ½, m = 2. Decoder uses MAP algorithm.

In order to calculate the number of errors after Soft Input Decryption and to compare it with the number of errors before the decryption, CCER (Cryptographic Check Error Rate) has been defined:

$$CCER = \frac{number\_of\_false\_cryptographic\_check\_values}{number\_of\_received\_cryptographic\_check\_values} \quad (4)$$

False cryptographic check values are cryptographic check values which were not verified after Soft Input Decryption.

CCER is presented in Fig.12 in relation to SNR. The correction has been done for trials up to the 8, i.e. 16 lowest absolute L-values (2^8 i.e. 2^{16} trials).

Decrypting gain [dB] means the improvement by Soft Input Decryption process compared to an increase of the SNR. For example, if 1 of 10 digital signatures cannot be verified (at 1 dB) without Soft Input Decryption, all cryptographic check values can be corrected.

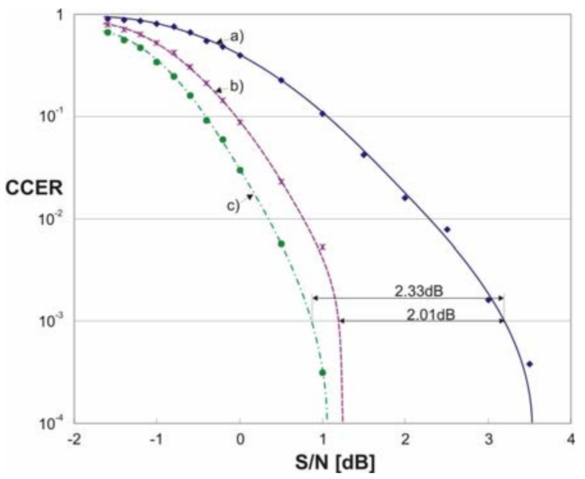

Fig.12 Decrypting gain of 321 bit cryptographic check value:

a) Convolutional Code without Soft Input Decryption
b) Convolutional Code with Soft Input Decryption using up to 8 lowest |L|-values
c) Convolutional Code with Soft Input Decryption using up to 16 lowest absolute |L|-values

8 Decrypting gain in Dependence on a Length of a Cryptographic Check Value

The following results are achieved for different lengths of cryptographic check values using convolutional coding.

Tests with 8, 16, 24 and 32 flipped bits (in Figures SID (8 bit), SID (16 bit), SID (24 bit) and SID (32 bit)) corresponding to the lowest positions of |*L*|-

values have been done for different lengths of cryptographic check values: 128, 160, 320 and 384 bits [ŽiRu07]. The results are presented in Fig. 13–16.

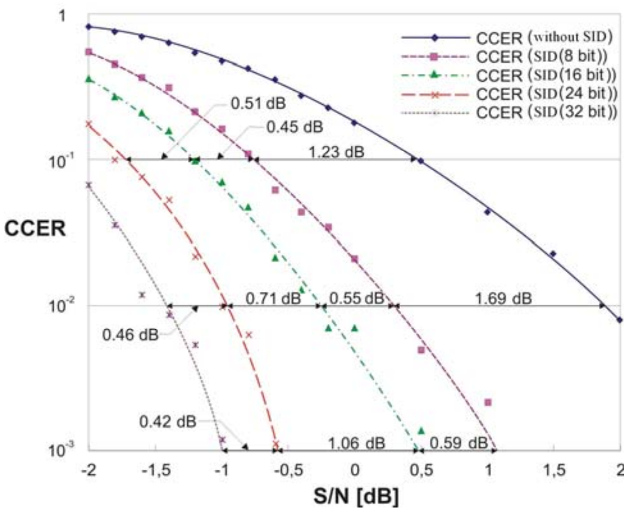

Fig.13 Decrypting gain of 128 bit cryptographic check

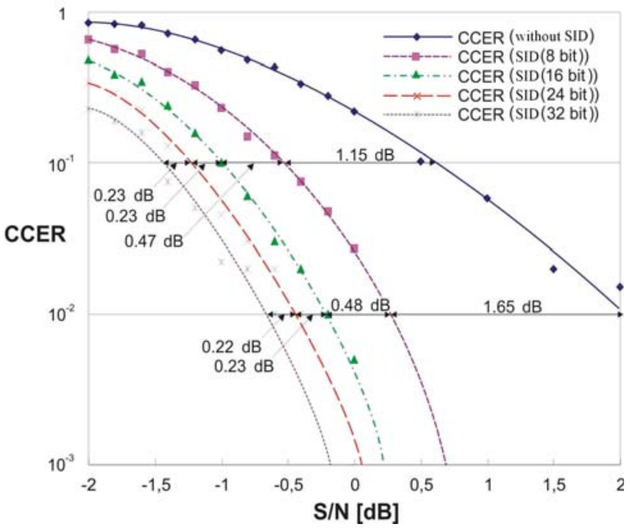

Fig.14 Decrypting gain of 160 bit cryptographic check value

As expected, the *CCER* after Soft Input Decryption is influenced by the length of cryptographic check values: it is higher when a cryptographic check values is longer. The reason is that longer cryptographic check values have more wrong bits distributed over a wider range of positions in the sequence of ordered |L|-values than shorter cryptographic check values, i.e. longer cryptographic check values need more combinations of bits which have to be changed to be corrected.

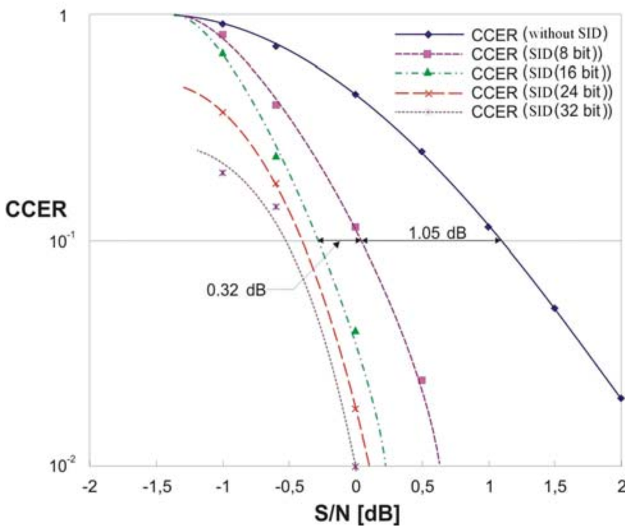

Fig.15 Decrypting gain of 320 bit cryptographic check value

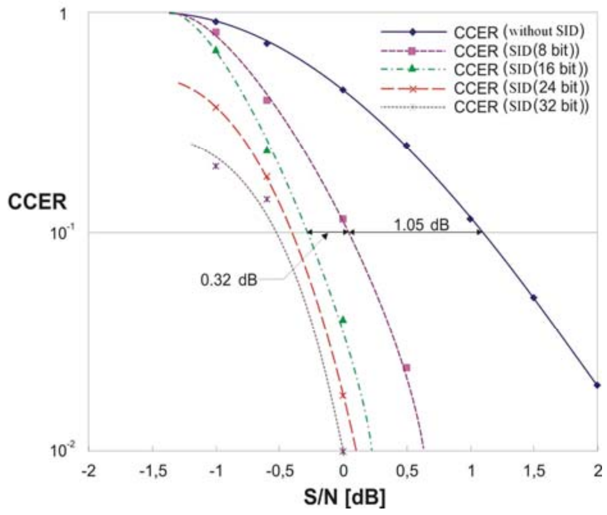

Fig.16 Decrypting gain of 384 bit cryptographic check value

9 Number of |L| - values verifications

The number of bits which are necessary to be changed in order to correct 95% of cryptographic check values has been obtained theoretically (chapter 9.1) and by simulations (chapter 9.2). Two questions arise:

How many bits have to be corrected depending on S/N, i.e. how many errors occur? The answer to

this question gives the theoretical number of verifications of absolute L-values.
How many L-values are necessary to correct errors which occur accordingly to the question 1? The answer to this question gives the number of verifications of |L|-values obtained by simulations.

The reason for the second question is that reliability values themselves are not reliable: 8 lowest |L|-values do not represent necessarily the 8 wrong bits (if 8 bits are wrong). Depending on the performance of channel decoding process, 8, 9, 10 or many more L-values may be necessary in order to detect wrong bits.

9.1 Theoretical Number of |L|-values Verifications

The answer to the question considers the theoretical number of |L|-values verifications, i.e. the number of bits which are wrong corrected of a length **n**.

The probability $\mathbf{P_{n,i}}$ that a code word of a length **n** – in this case a cryptographic check value 321 bits long – contains **i** errors is:

$$P_{n,i} = \binom{n}{i} p^i (1-p)^{n-i}, \tag{5}$$

where **p** represents BER.

Soft Input Decryption tests all possible combinations, that up to **k** bits are wrong, where **k** = 8 or 16 in this work (2^8 or 2^{16} combinations). That means that, after correction by Soft Input Decryption, only errors which are longer than **k** will not be corrected:

$$P_n = \sum_{i=k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{6}$$

In case that 95% errors have been corrected, i.e. in case when only 5% of errors remains, $P_n = 0.05$.

Calculation of **k** for correction of more than 95% errors for lengths of a cryptographic check value of length of 321 bits is presented in Fig.17 (dotted curve).

$$\begin{aligned} P_n = 0.05 \geq \sum_{i=k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} = \\ = 1 - \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned} \tag{7}$$

For **n** = 321:

$$P_{321} = 0.05 \geq 1 - \sum_{i=1}^{k} \binom{321}{i} p^i (1-p)^{321-i} \tag{8}$$

9.2 Tested Number of |L|-values Verifications

The number of bits which are necessary to be changed, in order to correct 95% of cryptographic check values, of length of 321 bits has been also shown in dependence on S/N in Fig.17 (solid curve). Simulations were done using convolutional codes.

The usage of up to 8 bits is sufficient for correction if S/N > 1 dB. For lower S/N the number of bits necessary for correction rapidly increases: for S/N < 0.5 dB 13 bits are needed for the correction. From Fig.17 it is shown that a number of bits which have to be changed increase almost exponentially with the decrease of S/N.

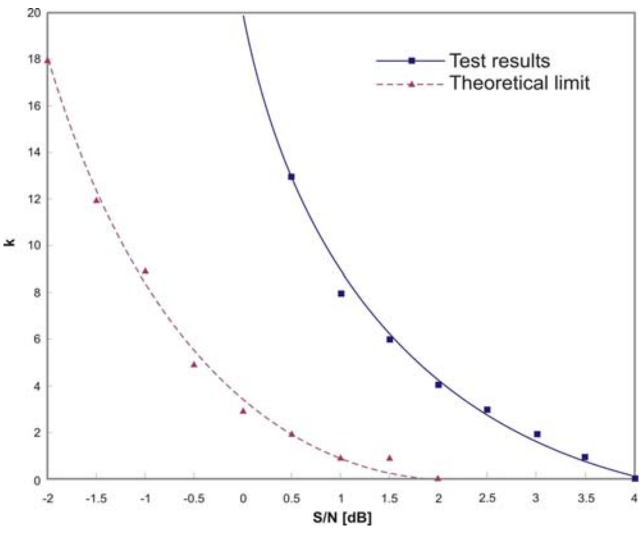

Fig.17 Number of changed bits for correction of 95 % of the cryptographic check values of a length of 321 bits

Comparing theoretical results and the results of tests from Fig.17, it is obvious that theoretical minimum of the number of bits that should be flipped in a SID process is much smaller then it is shown in tests. The reason for this difference is non-optimal choice of bits that have to be flipped, i.e. the “quality of distribution” of L-values is not good enough to give more precise information about positions of wrong bits. In case that better quality of L-values is provided (e.g. with the use of turbo decoding instead of convolutional decoding with MAP, and/or for cryptographic check values shorter than 321 bits), the better set of bits for flipping would be chosen, which leads to the smaller number of attempts needed for correction.

10 Conclusion

Integration of cryptography operations within decoding process is a promising way for its enhancement. A method based on soft decoding and cryptography (Soft Input Decryption) gives good results in improving the coding gain of cryptographic check values, with preserved security functions. Three possible applications of such a method are presented. It is shown that theoretical estimation of minimal number of attempts for correction of all wrong bits is different from the obtained results of simulation. This difference shows the possibility of further improvements of SID method, which would give even better results.

Future work should include optimization of the software realization of SID method. Optimization could be realized by improved mathematical solution of verification process, using the fact that each verification differs in only one bit from one of the previous verifications. Also techniques as genetic algorithms [16] for example, could be used for further software optimization. Realization of dynamic strategy and BER based strategy is another task of a future work, which promises faster and more efficient way of finding L-values of bits that should be corrected.

Future work should also concentrate on the possibilities to use cryptography as an enhancer of channel coding. Such cooperation between cryptographic and coding elements of the communication system introduces a principle of Joint Channel Coding and Cryptography.

References:

[1] N. Živić, C. Ruland, Softinput Decryption, *4th Turbocode Conference, 6th Source and Channel Code Conference, VDE/IEEE*, Munich, April 3 – 7, 2006.

[2] M. Bukowski, Turbo Codes in DSSS Systems and a Method of Their Performance Improvement in non-White Additive Noise, *Proc. Of the 6th WSEAS Int. Conference on Signal Processing, Robotics and Automation*, Corfu Island, Greece, February 16-19, 2007

[3] S.A. Barbulescu, *What a wonderful turbo world*, ISBN 0-9580520-0-X, Adelaide 2002.

[4] J. Hagenauer, L. Papke, Iterative decoding of binary block codes and convolutional codes, *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445, March 1996.

[5] F.-H. Huang, Evaluation of Soft Output Decoding for Turbo Codes, *thesis at the Faculty of the Virginia Polytechnic Institute*, May 1997, http://scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/

[6] J. Hagenauer, N. Dütsch, J. Barros, A. Schaefer, Incremental and Decremental Redundancy in Turbo Source-Channel Coding, *1st Int. Symposium on Control, Communications and Signal Processing*, Hammamet, Tunisia, pp. 595-598, March 2004.

[7] 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, pp. 284-287, March 1974.

[8] J. Hagenauer, P. Höher, A Viterbi algorithm with soft-decision outputs and its applications, *Proc. IEEE GLOBECOM `89*, Dallas, Texas, USA, vol. 3, pp. 1680-1686, November 1989.

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

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

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

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

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

[14] ISO/IEC 9797-2, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a hash-function*, 2000.

[15] ISO/IEC 15946-2, *Information technology – Security techniques – Part 2: Digital signatures*, 2002.

[16] I. Ivan, C. Boja, M. Vochin, I. Nitescu, C. Toma, M. Popa, Using Genetic Algorithms in Software Optimization, *Proc. Of the 6th WSEAS Int. Conference on Telecommunications and Informatics*, Dallas, USA, March 22-24, 2007