Feedback in Soft Input Decryption

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 , christoph.ruland@uni-siegen.de http://www.dcs.uni-siegen.de

Abstract: - This paper develops further the idea of use of Soft Input Decryption. The function of Soft Input Decryption is to integrate decryption into the decoding process. In this paper it is shown how Soft Input Decryption can be enhanced using feedback from the channel decoder. In this way, the cooperation between channel coding and cryptography is further develops. The effect of the feedback is explained on an example of the trellis diagram. Like in Soft Input Decryption, also in feedback method is the usage of reliability *L*-values of SISO channel decoder of great importance. Theoretical analysis, as well as results of computer simulations have been presented and discussed.

Key-Words: - Soft Input Decryption, feedback, L-values, cryptographic check values, MAP, SISO Channel Decoding, trellis

1 Introduction

In [1][2] the importance of using cryptographic elements - encryptor and decryptor has been shown. Normally, they are used in cryptographic operations to support communication security i.e. data integrity and authentication of data origin.

In [1-2], cryptographic elements are used in another way: as a 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 are used to correct cryptographic check values [1-2].

L-values values are valuable information about decoded bits, which are used in turbo decoding [3]. In [1-2] and in this work *L*-values are used in another way: as information to the next following entity – the decrypting mechanism.

Digital signatures [4][5][6], MAC [7]/H-MAC [8] ((Hash)-Message Authentication Code) and hash values [9] are used as cryptographic check values.

The main problem investigated in [1-2] and the proposed solution were:

- correction of cryptographic check values using *L*-values of channel decoding (solution: Soft Input Decryption)

The main problem investigated in this work and the proposed solution are:

- improving channel decoding using corrected cryptographic check values (solution: Soft Input Decryption using feedback).

2 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 [7]. 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 [7]. 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.

2.1 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 [9].

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 [9].

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.

2.2 Message Authentication Codes (MACs)

MAC is an application of a symmetric block cipher [7]. Examples of used block cipher algorithm are DES, 3–DES and AES.

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).

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 [7]:

- 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 [7].

2.3 Hashed Message Authentication Codes (H-MACs)

ISO/IEC 9797-2 [8] 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 [8].

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. 2.1).

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 [8]:

- 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 [8].

2.4 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.

There are two types of digital signatures:

1. signatures giving message recovery (Fig. 1) [6]
2. signatures with appendix (Fig. 2.17) [5].

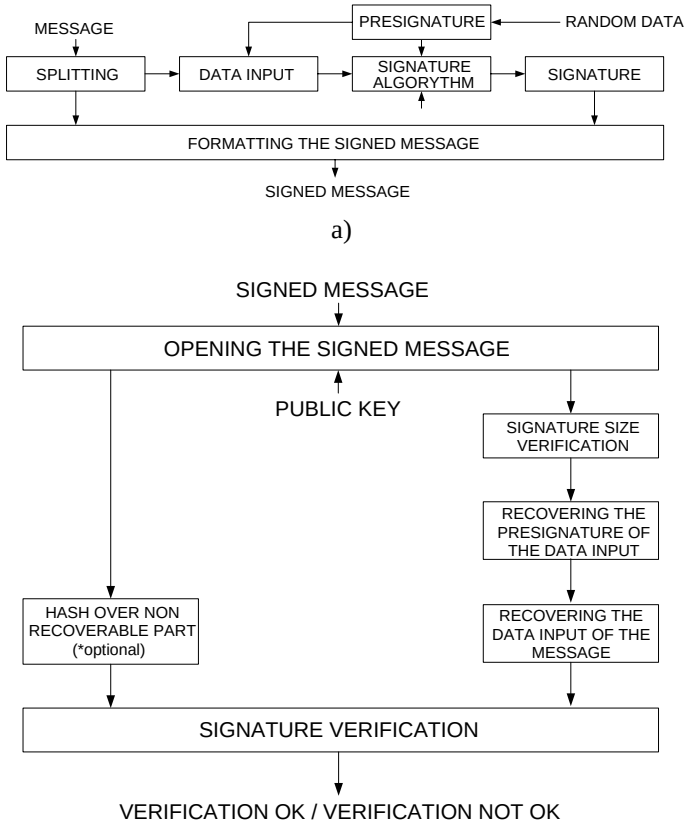

Fig. 1 Digital Signatures giving message recovery

A non-recoverable part which is covered by the hash value of the signature is optional. This option is not used in this work. Only messages which are included in the

signature and recoverable are considered.

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). In this work ECNR over GF(p) with a length of p of 160 bits is used, which results in a signature length of 320 bits.

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.

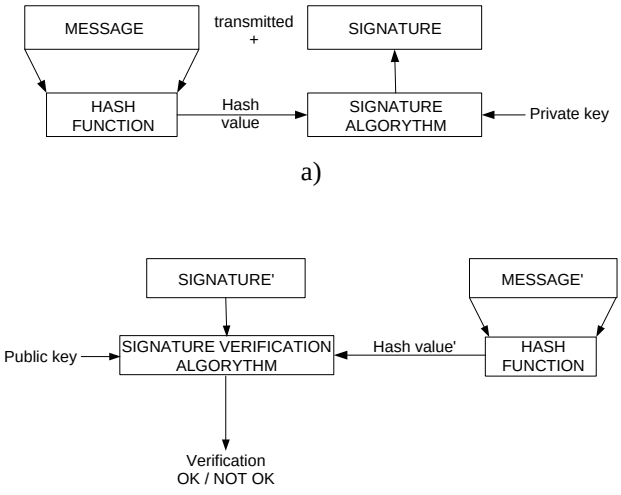

Fig. 2 Digital Signatures with appendix (simplified):

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.

3 SISO Convolutional Decoding

Convolutional codes are a type of error correcting codes which are very often used in wireless communication systems.

The convolutional encoder (5,7) (Fig.3) is implemented in this work because of its simplicity and often usage in theory and praxis.

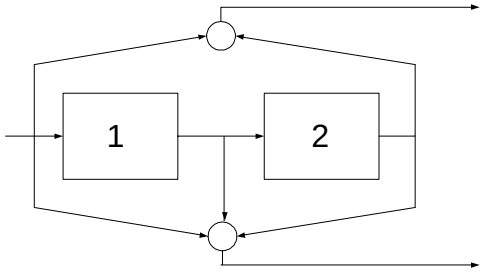

Fig. 3 Convolutional Encoder (r = ½, m = 2)

4 Soft Input Decryption

The strategy of Soft Input Decryption takes into account, that the security mechanism is successfully completed, if the used cryptographic check value is recognized by the decryptor to be correct (positive verification). If the verification is negative, the decryptor changes the bits with the lowest absolute L-values [10] and checks the result of the signature verification [2] (Fig. 4). This procedure takes place after each bit change or combination of bit changes as long as the verification is negative or a limit of computational effort is reached.

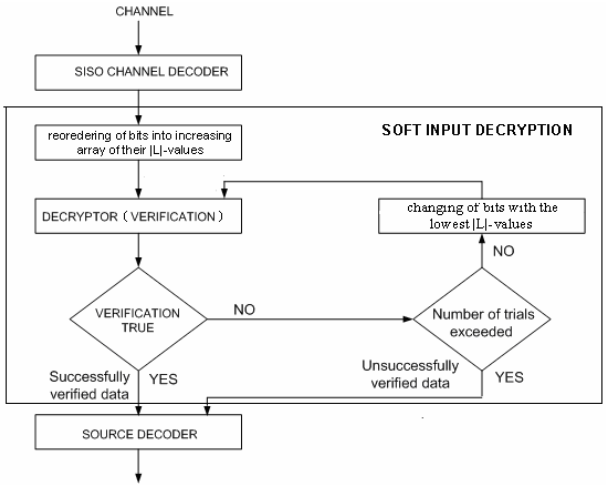

Fig. 4 Soft Input Decryption

In a case where the attempts for correction of the signature fail, because the signature has been modified intentionally (attack) or the number of errors is too large

as a result of a very noisy channel, the computational power is not sufficient to try enough combinations of flipping bits of low absolute *L*-values.

5 Introducing of the feedback into Soft Input Decryption

In chapter 4 it was explained how transmission errors can be corrected using Soft Input Decryption. This chapter explains how a corrected SID block can be used for improved error correction of channel decoding of another block by feedback method.

The output of the source encoder is a data block or data stream **u**. The stream **u** has to be authentic, which is realized by use of cryptographic check values. **u** is continuously split in two parts, which are called and considered as message *a* and message *b*. Each of both messages is extended by a cryptographic check value generated from the message using a cryptographic check function ***h*** (Fig.5):

ha = {ha_i} and **hb** = {hb_i}, i = 1, …, m.

Blocks *a* and *b* are the result of the concatenation of messages *a* and *b* with their cryptographic check values *ha* and *hb*:

$$a = a_1 a_2 ... a_{m_1} ha_1 ha_2 ... ha_{n_1} \quad (1)$$

$$b = b_1 b_2 ... b_{m_2} hb_1 hb_2 ... hb_{n_2} \quad (2)$$

For the case of simplicity, without limitation of generality, it is further assumed that $(m_2 + n_2)\ mod\ (m_1 + n_1) = 0$. Block *a* and block *b* form the joint message **v** (Fig.2):

$$v = \begin{cases} a_1 b_1 a_2 b_2 ... a_{m1} b_{m2} ha_1 hb_1 ha_2 hb_2 ... ha_{n_1} hb_{n_2}, \ if\ m_1 = m_2, n_1 = n_2 \\ a_1 b_1 ... \underbrace{b_{m_2}}_{m_1} a_2 ... a_{m1} \underbrace{b}_{m_2 - \frac{m_2}{m_1}+1} \underbrace{b_{m2}}_{n_1} ... ha_1 hb_1 ... \underbrace{hb_{n_2}}_{n_2 - \frac{n_2}{n_1}+1} ha_2 ... ha_{n_1} hb ... hb_{n_2} \\ if\ m_1 < m_2 \quad and \quad n_1 < n_2 \end{cases} \quad (3)$$

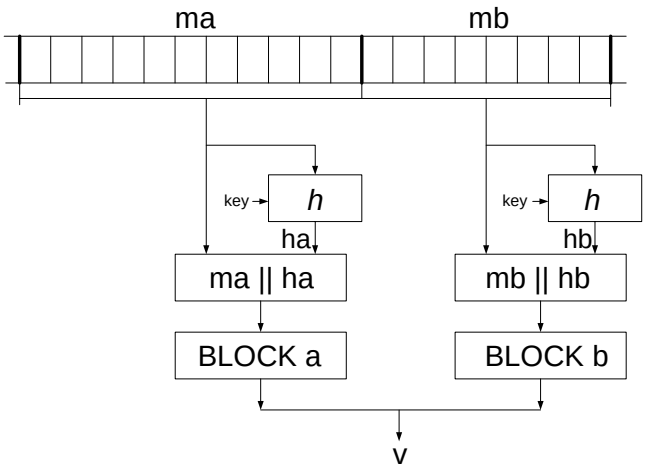

Fig. 5 Forming of a message ***v*** from a message ***u***

Let consider the case of $m_1 = m_2$ and $n_1 = n_2$, for the further simplicity, without lost of generality.

The algorithm of Soft Input Decryption using feedback is shown in Fig. 6.

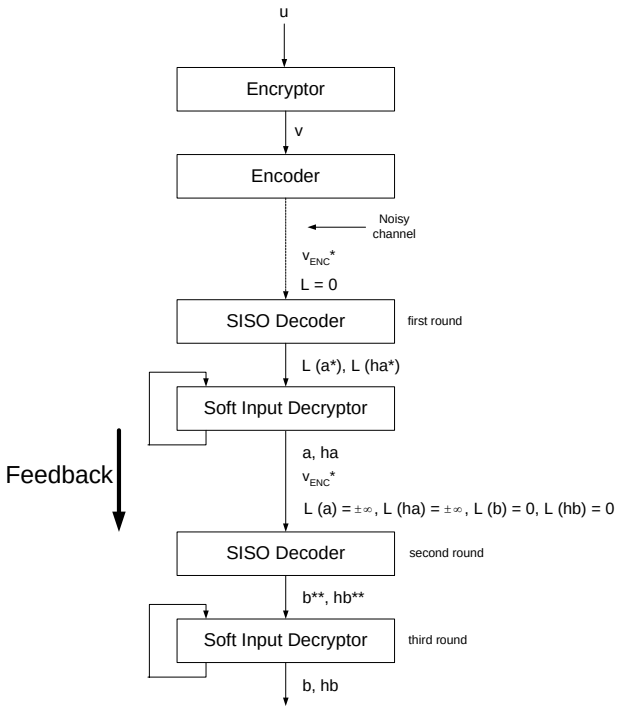

Fig. 6 Soft Input Decryption using feedback

The joint message **v** is encoded into $\mathbf{v_{ENC}}$ using a convolutional encoder (5,7) with r = ½ [11] (Fig.6):

$$v_{ENC} = a_{11} a_{12} b_{11} b_{12} ... a_{k1} a_{k2} b_{k1} b_{k2} \\ ha_{11} ha_{12} hb_{11} hb_{12} ... ha_{m1} ha_{m2} hb_{m1} hb_{m2} \quad (4)$$

After the transfer over the noisy channel, $\mathbf{v_{ENC}}$***** is received:

$$v_{ENC}* = a_{11}*a_{12}*b_{11}*b_{12}*...a_{k1}*a_{k2}* \\ b_{k1}*b_{k2}*ha_{11}*ha_{12}*hb_{11}*hb_{12}*...ha_{m1}* \\ ha_{m2}*hb_{m1}*hb_{m2}* \quad (5)$$

It should be emphasized that the values of the $\mathbf{v_{ENC}}$* are real numbers which are output values of the demodulator, and no binary values of $\mathbf{v_{ENC}}$ anymore!

Now the first round of the Soft Input Decryption with feedback starts: the SISO decoder decodes $\mathbf{v_{ENC}}$* and outputs *L*-values of **a*** and **ha*** to the Soft Input Decryptor. Based on these *L*-values, Soft Input Decryption tries to correct block *a* (**a** and **ha**). If Soft Input Decryption is successful, every second bit of **v** (i.e. **a** and **ha**) is corrected:

$$v^{(1)}* = a_1 b_1 *...a_k b_k * ha_1 hb_1 *...ha_m hb_m * \quad (6)$$

The *L*-values **L(a)** and **L(ha)** belong to the corrected bits, and they are set to ± ∞ (depending if the corrected bit is "0" or "1"). The *L*-values **L(b*)** and **L(hb*)** are set to the start values of 0.

In the second round of the Soft Input Decryption with feedback, the SISO decoder decodes the encoded data $\mathbf{v^*_{ENC}}$ and outputs the new values of **b**** and **hb**. The values of **a** and **ha** bits are already corrected in the first round, which "helps" the decoding algorithm to improve correction of **b** and **hb** bits. The BER after the second round is lower than after the first round, because of the lower error rate of **b** and **hb** bits. The second round represents a feedback of corrected *L*-values of block *a* in the first round to the SISO channel decoder.

In the third round of the Soft Input Decryption with feedback,, **b**** and **hb**** are further corrected by Soft Input Decryption. Because of the lower BER in comparison to the first round, Soft Input Decryption is more successful. That fact can be exploited by using a longer block *b* than of block *a* ($m_1 < m_2$ and $n_1 < n_2$).

6 Analysis of BER in Soft Input Decryption using feedback

The value of BER after each step of the algorithm is shown in Fig. 7.

Representation of collision probabilities and cryptographic check error rates of the algorithm in Fig. 7 is in "italic" form (P_{coll} and *CCER*), and of bit error rates in "non-italic" form (BER).

P_{coll} represents the probability that collision happen. Collisions happen if the calculated check value is equal to the received one, i.e. the verification is successful, although the received message or check value contains has been modified durino transmission. In case of simulations in this paper, collision can happen after the first verification (before the Soft Input Decryption starts) or by changing bits by Soft Input Decryption. P_{coll} can be calculated as [12]:

$$P_{coll} = \sum_{j=1}^{N}\sum_{i=0}^{j-1}\frac{n-(j-i)}{m+n-j}\frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}}\left[1-\left(1-\frac{2^{j-i}}{2^n}\right)^{2^{i-1}}\right]+ \\ \sum_{j=1}^{N}\sum_{i=1}^{j}\frac{m-i}{m+n-j}\frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}}\left[1-\left(1-\frac{2^{j-i}}{2^n}\right)^{2^{i-1}}\right]-\frac{1}{2^{m+n}} \quad (7)$$

xhere *m* is the length of the message and *n* is the length of the cryptographic check value. *N* is the maximal number of bits, which can be corrected (flipped) by Soft Input Decryption [1]. In {1],[2] and this paper, *N* = 16. *CCER* (Cryptographic Check Error Rate) is defined as:

$$CCER = \frac{number\ \ of\ \ incorrect\ \ SID\ \ blocks}{number\ \ of\ \ received\ \ SID\ \ blocks} \quad (8)$$

For example, if digital signatures are used as cryptographic check values:

$$CCER = \frac{number of\ \ incorrect digital\ signature}{number of\ \ received digital\ signature} \quad (10)$$

Number of incorrectly verified SID blocks = Number of not successfully verified SID blocks + Number of "wrong" verified SID blocks (with collision), because the "wrong" verification can happen in case of a collision. The complement of *CCER*:

$$\overline{CCER} = 1 - CCER = \\ = \frac{number of\ correct SID blocks}{number of\ received SID blocks} \quad (11)$$

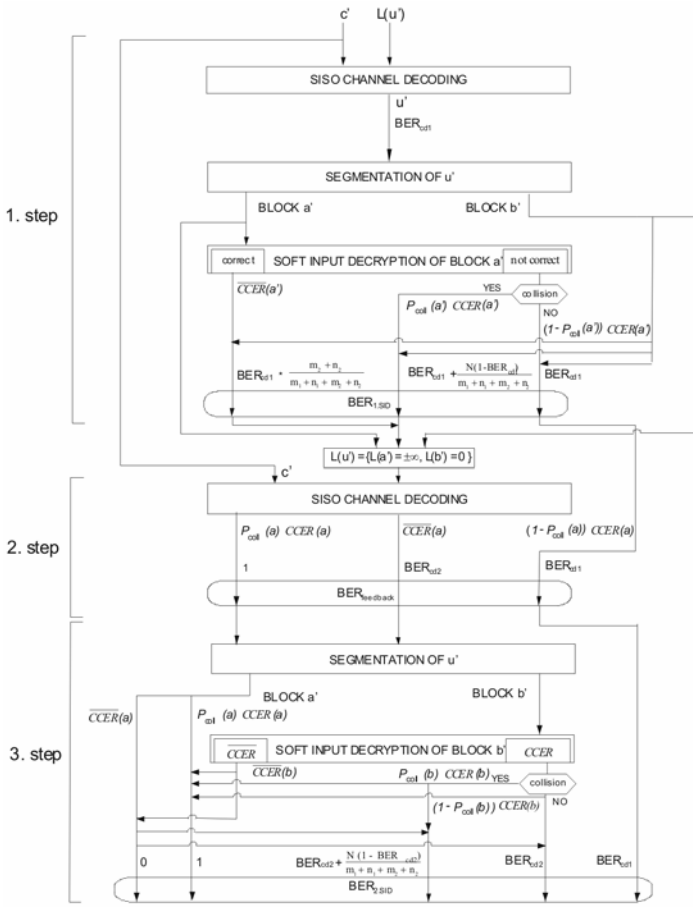

Fig.7 BER in Soft Input Decryption using feedback

BER is computed after each step of the algorithm. The probability of a collision in both blocks *a* and *b* is taken into account.

After the first Soft Input Decryption, three cases are possible:

1. the output of Soft Input Decryption gives a correct message The rate of correct blocks delivered by Soft Input Decryption is $\overline{CCER}$

2. Soft Input Decryption stopps because of finding a matching pair of the message and cryptographic check value, but the message is not the correct one. This happens with the collision probability P_{coll}. The bits which have not been flipped by Soft Input Decryption have the error rate BER_{cd1}. The flipped bits, whose number is smaller than *N*, are assumed to be wrong (the worst case). When these blocks are fed back to channel decoder, it is assumed that $BER_{cd2} = 1$ (the worst case). The rate of these blocks is $P_{coll} \cdot CCER$.

3. Soft Input Decryption stopped by exceeding the limit of number of trials. The rate of not corrected blocks is $(1 - P_{coll}) \cdot CCER$.

The sum of probabilities of all three cases is 1:

$$\overline{CCER} + (1 - P_{coll}) \cdot CCER + P_{coll} \cdot CCER = 1 \qquad (12)$$

After the second Soft Input Decryption, nine cases are possible with following probabilities:

1. block *a* and block *b* are correct:
$\overline{CCER}(a') \cdot \overline{CCER}(b')$

2. block *a* is correct and block *b* is not correct, because a collision happened:
$\overline{CCER}(a') \cdot CCER(b') \cdot P_{coll}(b)$

3. block *a* is correct and block *b* is not corrected after exceeding the number of trials:
$\overline{CCER}(a') \cdot CCER(b') \cdot (1 - P_{coll}(b))$

4. block *a* is not correct, because a collision happened, and block *b* is correct:
$CCER(a') \cdot P_{coll}(a) \cdot \overline{CCER}(b')$

5. block *a* is not correct, because a collision happened, and block *b* is not correct also because of a collision:
$CCER(a') \cdot P_{coll}(a) \cdot CCER(b') \cdot P_{coll}(b)$

6. block *a* is not correct, because a collision happened, and block *b* is not corrected after exceeding the number of trials:
$CCER(a') \cdot P_{coll}(a) \cdot CCER(b') \cdot (1 - P_{coll}(b))$

7. block *a* is not corrected after exceeding a number of trials and block *b* is correct:
$CCER(a') \cdot (1 - P_{coll}(a)) \cdot \overline{CCER}(b')$

8. block *a* is not corrected after exceeding a number of trials and block *b* is not correct because a collision happened:
$CCER(a') \cdot (1 - P_{coll}(a)) \cdot CCER(b') \cdot P_{coll}(b)$

9. block *a* is not corrected after exceeding a number of trials and block *b* is also not corrected after exceeding a number of trials:
$CCER(a') \cdot (1 - P_{coll}(a)) \cdot CCER(b') \cdot (1 - P_{coll}(b))$.

In case 4, 5 and 6, $BER_{2.SID} = 1$ (the worst case), because the feedback of a wrong block results in a complete wrong block *u* (worst case). The probability that case 4, 5 or 6 happens, is:

$$CCER(a') \cdot P_{coll}(a) \cdot [\overline{CCER}(b') + CCER(b') \cdot P_{coll}(b) + CCER(b') \cdot (1 - P_{coll}(b))] = CCER(a') \cdot P_{coll}(a) \quad (12)$$

In case 7, 8 and 9 block *a* could not be corrected and therefore there is no feedback and no second Soft Input Decryption. For that reason, the algorithm stops and $BER_{2.SID} = BER_{cd1}$.

The probability that case 7, 8 or 9 happens, is:

$$CCER(a') \cdot (1 - P_{coll}(a)) \cdot [\overline{CCER}(b') + CCER(b') \cdot P_{coll}(b) + CCER(b') \cdot (1 - P_{coll}(b))] = CCER(a') \cdot (1 - P_{coll}(a)) \quad (13)$$

The sum of probabilities of cases 4, 5, 6 and 7, 8, 9 is than *CCER(a')*. As the sum of probabilities of cases 1, 2, 3 is equal to $\overline{CCER}(a')$, the probability of all 9 possible cases is 1, as those cases are elements of a complete event.

BER after each of three steps of algorithm is presented by following equations:

$$BER_{1.SID} = \overline{CCER}(a') \cdot BER_{cd1} \cdot \frac{m_2 + n_2}{m_1 + n_1 + m_2 + n_2} + CCER(a') \cdot P_{coll}(a) \cdot (BER_{cd1} + \frac{N \cdot (1 - BER_{cd1})}{m_1 + n_1 + m_2 + n_2}) + CCER(a') \cdot (1 - P_{coll}(a)) \cdot BER_{cd1} \quad (14)$$

$$BER_{feedback} = \overline{CCER}(a') \cdot BER_{cd2} + CCER(a') \cdot P_{coll}(a) \cdot 1 + CCER(a') \cdot (1 - P_{coll}(a)) \cdot BER_{cd1} \quad (15)$$

$$BER_{2.SID} = 0 + CCER(a') \cdot P_{coll}(a) \cdot 1 + CCER(a') \cdot (1 - P_{coll}(a)) \cdot BER_{cd1} + \overline{CCER}(a') \cdot CCER(b') \cdot P_{coll}(b)) \cdot (BER_{cd2} + \frac{N \cdot (1 - BER_{cd2})}{m_1 + n_1 + m_2 + n_2}) + \overline{CCER}(a') \cdot CCER(b') \cdot (1 - P_{coll}(b)) \cdot BER_{cd2} \quad (16)$$

7 Influence of the feedback to BER

L-values of corrected bits of block *a* ($L = \pm \infty$) are feedback information sent to the channel decoder, enabling better decoding results of block *b*. In this way the probability decreases, that parts of the trellis are wrong, and thereby some bits are wrong decoded.

Fig. 8 shows an example of correcting of block *b* (in the second round), if block *a* is already corrected (in the first round). Each transition on the trellis shows the value of the sent bit. For example, two transition paths (between the states s_{k-2} and s_{k-1}, and s_k and s_{k+1}) which are "bold" in Fig. 8, show that the sent bit was "1" in both cases. If these transition paths belong to corrected bits of block *a*, it is easy to find the transition between the states s_{k-1} and s_k, which connects both of paths (the bit of block *b*). The resulting "bold" path shows that the sent bit was "0". In this way, bits of block *a* enable correction of decoded bits of block *b*.

By knowing some of transition paths in the trellis, correction of the rest of transition paths is enhanced, because the number of possible transitions connecting known transition paths decreases. This fact is used for feedback from Soft Input Decryption to the SISO convolutional decoder in this paper.

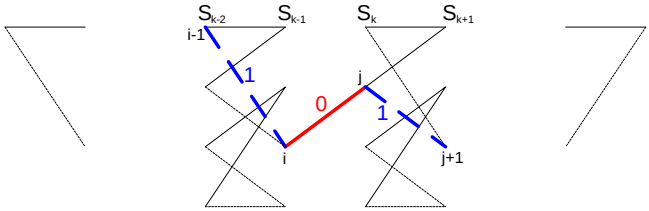

Fig.8 Trellis diagram with 2 known and 1 unknown transition

Example:

- input of encoder: **1 0 1** ($a_1 = 1$, $b_1 = 0$, $a_2 = 1$);
- input of decoder: **00 00 00**;
- encoder in Fig. 3 has been used (state and trellis diagram of the encoder are in Fig. 9 and Fig. 10 respectively);
- after successful Soft Input Decryption of bits a_1 and a_2, their *L*-values are set to:

$L(a_1) = L(a_2) = -\infty$

- b_1 = ?

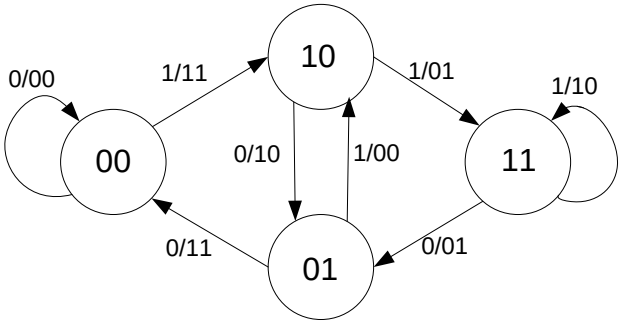

Fig.9 State diagram of the used encoder

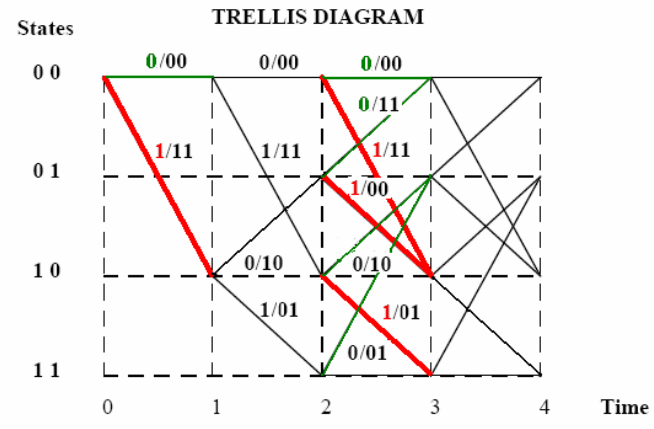

Fig.10 Trellis diagram of the used encoder

As it is known, the first bit is "1" (a_1 = 1), the only possible transition in the trellis is the bold one ("1/11"). Two transition paths are possible from this bold transition branch in the trellis: "0/10" and "1/01". As it is also known that the third bit is "1" (a_3 = 1), there are three possible transitions in the trellis which accord a_3 (bold): "1/11", "1/00" and "1/01". There is only one path left which connects one of these three paths and the first bold path "1/11": the path "0/10". Evidentially, the received bits 00 00 00 are wrong: they should be 11 10 00, and b_1 has to be "0".

In this way, the possibilities of wrong decisions are decreased: from four possible transition paths, the possibility of a wrong decision of the second sent bit decreases to two paths, knowing that a_1 = 1; then, this number of possible transition paths is reduced to 1, knowing that a_2 = 1.

Similarly, decoding continues finding new possible paths through the trellis, with higher possibility of finding the right path. Therefore, BER decreases, i.e. MAP decoding results improve.

8 Results of Simulations

Experiments are performed with the convolutional encoder (5,7) with r=1/2. BPSK modulation, AWGN channel and MAP [13] decoder are used. In all simulations cryptographic check values are used, which fulfill security criteria by their lengths. For that reason, collision probability is negligible, i.e. $P_{coll} \approx 0$.

8.1 Block *a* and *b* of the same length

In the following example, $m_1 = m_2 = n_1 = n_2 = 80$. Simulations have been performed in C/C++ programming language. For each point of curves, 50000 simulations have been performed, which is more than enough for getting 99 % reliable results [14].

BER after each step of the algorithm is calculated and shown in Fig. 11.

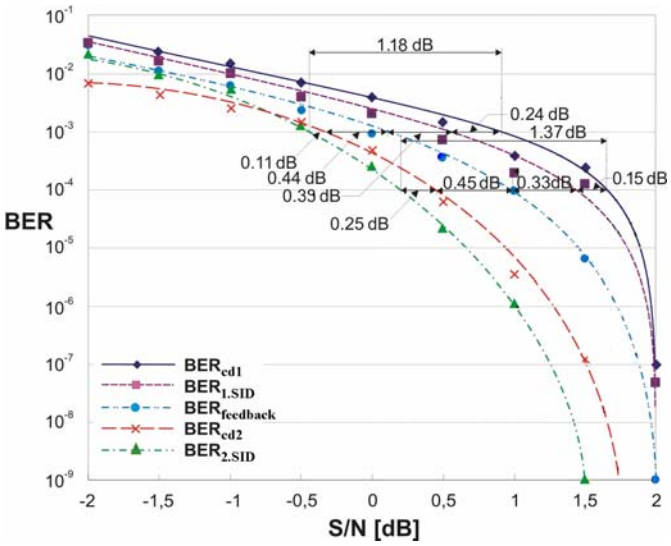

Fig.11 BER after each round of Soft Input Decryption using feedback

It is obvious that each round enhances the efficiency of the algorithm of Soft Input Decryption using feedback, i.e. BER after each round decreases.

Fig. 8 shows that the total coding gain (after the third round, in comparison to convolutional decoding) of Soft Input Decryption using feedback varies, dependant on S/N. For higher S/N, total coding gain is above 1 dB (BER ~ 10^{-4}, 10^{-5}).

Coding gains obtained by simulations confirm theoretical results from chapter 6.

8.2 Block *b* longer then block *a*

In the following example, block *b* is three times longer than block *a*: $m_1 + n_1 = 160$ ($m_1 = n_1 = 80$) and $m_2 + n_2 = 480$ ($m_2 = 400$ and $n_2 = 80$).

All conditions and characteristics of simulations are the same as in 8.1. BER after each step of the algorithm is calculated and shown in Fig. 12.

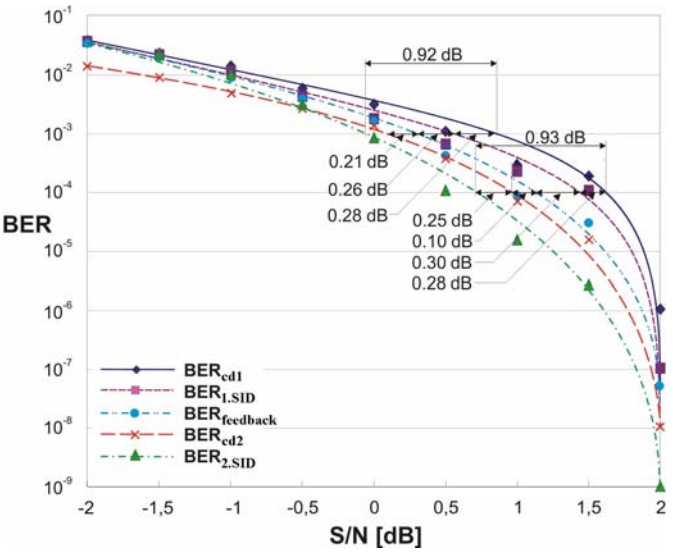

Fig.12 BER after each round of Soft Input Decryption using feedback

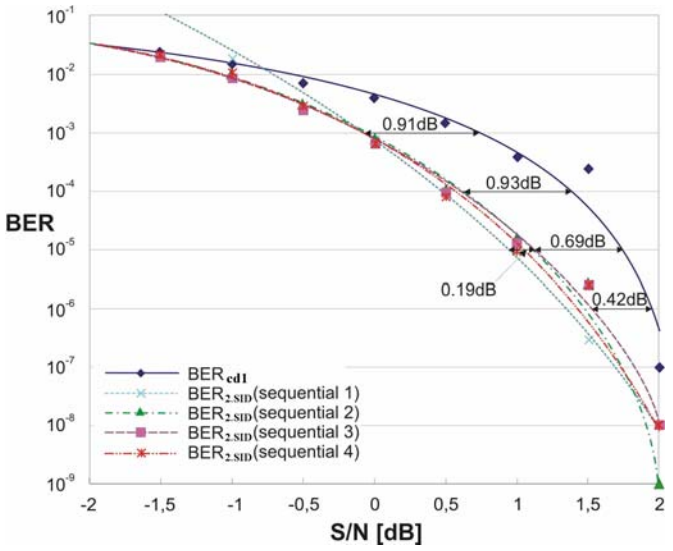

Fig.13 Coding different lengths of block *a* and block *b* in comparison to channel decoding

In Fig. 12 it can be seen that the coding gain grows with increasing *S/N*. For example, for *S/N* = 2 dB, $BER_{1.SID}$ is around 10^{-6}, but after feedback method BER decreases under 10^{-7}. The improvement accomplished by the use of Soft Input Decryption of block *b* is shown by $BER_{2.SID}$, i.e. by coding gain of the Soft Input Decryption of block *b*. It is obvious that the influence of the Soft Input Decryption of block *b* is significant. For example, for BER of 10^{-4}, the coding gain of the feedback is 0.58 dB and the coding gain after the 2. Soft Input Decryption is 0.93 dB.

Coding gains obtained by simulations confirm theoretical results from chapter 6.

9 Influence of blocks lengths to BER

Following simulations of Soft Input Decryption using feedback examine influence of constant length of *u* with various lengths of blocks *a* and *b* on coding gains. Coding gains for length of **u** of 640 bits is shown in Fig.13 in comparison to channel decoding (BER_{cd1}). The message *u* is divided into block *a* and block *b* with the lengths given in Table 1.

Sequential test No	Length of block *a*	Length of block *b*
1	128	512
2	160	480
3	212	480
4	320	320

Table 1 Lengths of blocks *a* and *b*

The results in Fig.13 show no significant difference between BER for different lengths of *a* and *b* blocks, i.e. the results are not very much dependant on lengths of individual blocks *a* and *b* for the constant length of a message **u**. The reason for that is that advantage of shorter block *a* (better results of Soft Input Decryption) is neutralized by disadvantage of use of longer block *b* (worse results of Soft Input Decryption) and vice versa.

Coding gains obtained by simulations confirm theoretical results from chapter 6.

10 Conclusion and the future work

This paper extends Soft Input Decryption by introducing feedback which improves SISO channel decoding, using results of Soft Input Decryption. The algorithm of Soft Input Decryption consists of three rounds. The result of each round is an additional coding gain. The total coding gain is above 4 dB for high S/N ratio.

Future work should include optimization of the software realization of SID method. Techniques as genetic algorithms [15] for example, could be used for further software optimization.

The future work should include the analysis of the influence of an extension of block *b* of the data **u**, as the successful correction of block *a* enables better decoding of the next decoding round. The corrected bits should not be limited to every second bit, but the distance between corrected bits should be extended.

If the message is divided into several (not only two) parts, Soft Input Decryption using feedback becomes an iterative method. In this way it is possible to correct a data stream which is not limited by its length.

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] N. Živić, C. Ruland, Channel Coding as a Cryptography Enhancer, *11th WSEAS Int.Multiconference*, Agios Nikolaos, Crete Island, Greece, July 23-28, 2007.

[3] C. Berrou, A. Glavieux, P. Thitimajshima: Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes, *Proc. IEEE International Conference on Communication*, Geneva, Switzerland, vol. 2/3, pp. 1064-1070, 1993

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

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

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

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

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

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

[10] D. Chase: A Class of Algorithms for Decoding Block Codes with Channel Measurement Information, *IEEE Trans. Inform. Theory*, IT-18, pp. 170-182, January 1972

[11] Drajić, D.B.: Uvod u teoriju informacija i kodovanje, *Akademska misao*, Beograd, 2004

[12] N. Živić, C. Ruland, Collisions in Soft Input Decryption, *WSEAS Int.Multiconference, American Conference on Applied Mathematics*, Harvard University, Harvard, USA, March 24-26, 2008

[13] Bahl, L., Jelinek, J., Raviv, J., Raviv, F.: Optimal decoding of linear codes for minimizing symbol error rate, *IEEE Transactions on Information Theory*, IT-20, pp. 284-287, March 1974.

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

[15] 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