The performance of error correcting codes in discrete and continuous fuzzy pairing protocols

Henrique de Rocha Deus[1], Vladimir Portela Parente[2], Jeroen van de Graaf[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

[2]Graduate Program in Electrical Engineering
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

{henriquerdd,jvdg}@dcc.ufmg.br, vladimir@cpdee.ufmg.br

Abstract. *In fuzzy pairing, two parties compare two bit strings which are supposed to be similar, but almost never identical. So A and B engage in a protocol to verify whether $d(s_A, s_B)$ is less than some threshold T; if not, the parties abort and authentication failed. Here $d()$ is to be taken as the Hamming distance. One standard protocol is the code-offset method: A computes a random vector x such that $s_A - x$ is a code word of some pre-agreed error-correcting code and sends x to B, who decodes $s_B - x$. Together they verify whether the two decoded codewords are the same. A common secret key can be obtained subsequently,*
We cast this problem in a different context, in which A and B want to compare continuous *signals, instead of discrete bit strings. We test the code offset method for four classes of error-correcting codes: Reed-Solomon (RS) Codes, Low Density Parity Check (LDPC) Codes, Repeat-Accumulate Codes (RAC) and Low Density Lattice Codes (LDLC).*
For similar error correction capability our results show that RS codes perform slowly, while LDPC and RAC which very similar are both really fast. LDLC has the best correction capabilities, but are slower because of their mathematical complexity. Our results can be generalized to fuzzy extractors.

1. Introduction

1.1. Fuzzy pairing with bit strings

Consider the following setting: Alice has a string s_A while Bob has a string s_B. Alice and Bob know that their strings are similar, but probably not identical. In addition, they know or suspect that the adversary Eve has some information about their bit string, but up to some known limit. In addition we assume that Alice and Bob can send each other authenticated messages, to which Eve also has access. Now consider the following cryptographic functionalities:

Fuzzy mutual authentication: A and B want to verify whether $d(s_A, s_B)$ is less than some threshold T, where $d()$ is to be taken as the Hamming distance. If true, authentication succeeds; if not, the parties abort and this authentication attempt failed.

Fuzzy key establishment: Alice and Bob want to obtain two smaller strings s'_A and s'_B such that $s'_A = s'_B$ with overwhelming probability.
This problem is otherwise known as key reconciliation or information reconciliation, and seems to have occurred first in the context of quantum cryptography: the bit strings are the result of those positions in which Alice and Bob used the same basis when transmitting resp. measuring the quantum state.
Note also that after reconciliation succeeded, Eve still might have information about the common string $s' = s'_A = s'_B$, so it is typically followed by another protocol known as privacy amplification. See [Bennett et al. 1995] for details.

Fuzzy pairing: In fact, pairing is the combination of the two preceding functionalities: the two parties decide to trust each other, and agree on some symmetric key to establish a secure communication channel.

Note that there exist a simple protocol to verify equality at the sacrifice of one bit per round: Alice choose random $r \in \mathbb{Z}_2^m$ and computes the parity (xor sum) $p_A = par(s'_A \wedge r)$, and sends the pair (r, p_A) to Bob. Bob computes $p_B = par(s'_B \wedge r)$. It is easy to see that if $s'_A = s'_B$ then $p_A = p_B$ always, whereas if $s'_A \neq s'_B$ then $p_A \neq p_B$ with probability $\frac{1}{2}$. So this protocol, which we call EQUAL must be repeated a sufficient number of times.

1.2. The code-offset method

The more famous reconciliation protocols for bitstrings, such as [Brassard and Salvail 1993], rely heavily on interaction. But a simpler approach can be based on error correcting codes. Later known as the *code-offset method*, $\mathcal{A}$ computes a random vector $x \in \mathbb{Z}_2^n$ such that $s'_A - x$ belongs to $C \subseteq \mathbb{Z}_2^n$, some pre-agreed error-correcting code, and sends x to B, who decodes $s'_B - x$. Then the parties verify whether the decoded codewords are equal, using the EQUAL protocol outlined above. If C is a linear code, an equivalent result is obtained if A sends the syndrome of s'_A to B.

1.3. Fuzzy pairing with signals

Traditionally, the information to be reconciled is represented as bits. In this paper we discuss a different setting: the information shared between A and B are signals that are similar, but not identical. In principle these signals can be of any kind: audio, wifi signal strength, accelerometer data, biometric data etc.

To model this setting adequately we need a continuous communication model. The natural choice is the AWGN (Additive White Gaussian Noise) channel, which is essentially as follows: imagine two Gaussian probability distributions: $N_0 = N(-1, \sigma)$ with center 0 and $N_1 = N(-1, \sigma)$ with center 1. To send a bit v, A produces a signal according to N_v; this signal is received by B as w. B's task is to guess v given w. Obviously, B's error depends on σ. We also need to substitute the Hamming distance for the Euclidean distance in $\mathbb{R}^n$.

In order to reconcile information in this continuous context, the logical step is to apply the code-offset method for an error-correcting code suitable for this context. In fact, no other approach seems possible since all the interactive protocols do not seem to carry over to this new setting.

1.4. Our research

It turns out that in all realistic scenarios for fuzzy pairing, the main performance bottleneck is decoding time, since it translates directly in the time the users have to wait to know if the pairing process succeeded or not.

In this paper we analyse the decoding performance in the signal (i.e. continuous) setting with AWGN channel for four classes of error correcting codes: Reed-Solomon (RS) Codes, Low Density Parity Check (LDPC) Codes, Repeat-Accumulate Codes (RAC) and Low Density Lattice Codes (LDLC).

This choice of these codes is natural: RS are very popular and well-known. Later LDPC codes were developed with superior performance; RAC were included because they are a dramatic simplification of LDPC with respect to encoding, while using essentially the same decoding algorithm. Finally, in theory LDLC will correct more than the others, however due to its complexity it is expected to be much slower. In this work we want to study how big that difference in both correction and speed is in practice.

1.5. Comparison with other work

Many existing pairing protocols, including Bluetooth, are based on the Diffie-Hellman Key Exchange. This has two disadvantages. First, it is based on computational assumptions. Second, to exclude a man-in-the-middle attack some user intervention is necessary. By using the newest features of smart phones, this comparison task can be performed by one of the devices. The hash value is no longer communicated or compared by a human, but through some alternative, reliable channel with the help of a human. Possibilities are:

- One display shows a code, user enters it in the keyboard of the other device.
- Printer prints or displays a bar code; a camera takes a photo.
- One smart phone plays a random melody; the other records the sound.

See [Kumar et al. 2009] for an overview. Observe that all these options require user action.

In our fuzzy pairing protocol no human intervention is necessary. Also it is not based on a computational assumption but on physical proximity: we assume that $\mathcal{A}$ and $\mathcal{B}$ have access to some similar signal, whereas the signal that Eve can obtain is not sufficiently similar.

1.6. Paper outline

The next section present a broad overview of fuzzy cryptography, providing many additional references to reconciliation and fuzzy crypto protocols. In section 3 we outline our protocol for the audio setting, and discuss the literature on other approaches. Section 4 discusses the classes of codes we tested. In section 5 we present our simulation and results, while Section 6 contains the conclusion.

2. Fuzzy Mutual Authentication

Fuzzy cryptography can be described as the area in which legitimate parties have similar but not identical information. Historically this problem has been approached in two different ways: one is based on information reconciliation, the other is called fuzzy cryptography.

2.1. Reconciliation protocols

A great variety of quantum key distribution protocols make the legitimate parties to have keys with little difference because of eavesdroppers and imperfections in the transmission line and detectors. Therefore, we can see this problem as Alice transmitting the key through a binary symmetric channel for Bob. In order to Alice and Bob have the same key, an error correction has to be done.

[Brassard and Salvail 1994] describes the Cascade protocol. It takes two correlated strings from Alice and Bob, divide them in blocks and, in each round, sends the parity of a block. Essentially, a binary search is done in order to find a different bit. However, because the parity of some blocks are revealed, privacy amplification has to be executed after the protocol to reduce the adversary's knowledge. This is one of the first works to address the key exchange problem when the legitimate parties have some correlated knowledge.

[Linnartz and Tuyls 2003] proposed an authentication scheme that can also be slightly modified to create a key conciliation protocol. Additional references are [Buhan et al. 2007a][Chang et al. 2004][Zheng et al. 2006][Verbitskiy et al. 2010].

2.2. Fuzzy cryptography constructions

[Juels and Sudan 2006] proposes a new construction called fuzzy vault. If Alice has a secret k and a set A, she does a commitment with the secret k. Bob can only perform a decommitment if he has a set B close enough to A. A practical protocol using Reed-Solomon codes (described in [Reed and Solomon 1960]) is showed.

In [Dodis et al. 2008] two important constructions are showed. The secure sketch construction can be seen as an authentication scheme. One side is the user and the other is the database with the user information. Authentication is only possible if the user has a feature that is close enough by some metric to the information that the database has. The other construction is the fuzzy extractor. It allows that Alice and Bob distill a secret and common key if they both have information with enough correlation.

[Buhan et al. 2007b] extends the definition of fuzzy extractor to continuous sources. It shows that it is possible to see the schemes that work with continuous source as authentication systems with some False Acceptance Rate (FAR) and some False Rejection Rate (FRR) with a public distribution of the feature of the users (global distribution) and private distributions of the users (each user has his own).

In [Verbitskiy et al. 2010] the results regarding the cs-fuzzy extractor (fuzzy extractor for continuous source) are extended and some limits of security are calculated. It is important to highlight that has been discussed that no universal fuzzy extractor for continuous source can exist. It is necessary to make assumptions about the distribution used.

3. Fuzzy pairing based on audio

We illustrate the context with a specific scenario in which we use audio as the common signal between two devices (smartphones) A and B. This was indeed the original context of our research, after we discarded accelerometer data for having insufficient entropy. We make the following assumptions for our protocol:

(a) Each device has a microphone.
(b) Each device has processing capacity.
(c) The devices dispose of a wireless, unauthenticated communication channel. An adversary can listen to this channel, but cannot modify messages sent by A or B.
(d) The two microphones are physically close to each other during the recording phase in order to have very similar sound recordings. We call this the *private sound zone*.
(e) An adversary cannot enter the private sound zone of the two legitimate parties without being detected, and if he tries, A and B will simply not execute the protocol. Therefore the adversary cannot obtain information about the audio recording.

Fuzzy pairing protocol:

1. A and B share the same sound environment and are synchronized in some way not discussed here. They record the sound.
2. Each party submits the collected audio data to an identical set of transformations resulting in two singular vectors s_A and s_B.
3. $\mathcal{A}$ computes a random vector $x \in \mathbb{Z}_2^n$ such that $s_A - x$ belongs to $C \subseteq \mathbb{Z}_2^n$, some pre-agreed error-correcting code, and sends x to B.
4. A decodes $s_A - x$ to obtain s'_A while B decodes $s_B - x$ to obtain s'_B. This is the most time-consuming protocol step. Observe that if $d(s_A, s_B) < d_{min}(C)$ then $s'_A = s'_B$.
5. A and B both perform privacy amplification, as follows. A compute $s''_A = h(s'_A)$, where $h()$ is a universal hash function. B applies h to his value.
6. A and B use the EQUAL protocol to test whether $s''_A = s''_B$. If true, authentication succeeded and s'' is there common shared secret key. If not, authentication has failed and they abort

3.1. Other authentication protocols using audio

[Schurmann and Sigg 2013] proposed and built an application using a fuzzy extractor scheme. Two computers, each one with it own microphone, agree on a common time to record. The audio data was processed in order to generate a fingerprint of 512 bits. A secret random string of 204 bits was generated and encoded to 512 bits using a Reed-Solomon code, specifically $RS(2^{10}, 204, 512)$. This value was added to the audio fingerprint and then transmitted to the other legitimate part.

One positive aspect of this work is the amount of tests that had been done. With the analysis of four environments and varying the distance from the microphones, 7500 tests were done. This work has the same goal and uses the same source to extract features. Furthermore, the audio processing uses STFT to calculate the energy difference between frames. Although this approach works with the discrete Hamming distance, the STFT processing is an interesting choice to use with audio.

[Goodrich et al. 2006] creates a system called Loud and Clear (L&C). It users that fact that every device, such as a smartphone, has at least one user. A sound is produced at a device and the user needs to verify if the sentence recognized is the same. From this, one-way authentication is possible (two-way is just another execution of the protocol). Although this protocol uses sound in order to authenticate the devices, it ignores the sound context of the surrounding of both devices. Furthermore, the improvement suggested in

the paper uses the Diffie-Hellman key exchange protocol. Summarizing, it returns to the computational security in an unnecessary way. Albeit the protocol is interesting, it is not suitable for actual use.

4. Fuzzy authentication using other phenomena

4.1. Authentication with Acceleration

[Kirovski et al. 2007a] and [Kirovski et al. 2007b] proposed a key conciliation protocol based on acceleration. Before doing the readings, it is necessary to pair the device and synchronize the clocks in order to have correlated information. Next, a shaking of the devices is done. The name of this work comes from the fact that shaking the two devices is similar to shake a Martini. The readings are separated in quantization zones and each zone is divided between an correct zone and an error zone. The preliminary information is if the measurement is in the error zone or not. Alice and Bob reveal their preliminary information. If they agree on a particular point, then the quantization zone where the measurement belongs is kept as private information. If they do not agree, then they discard the information.

With this approach, is possible that Alice and Bob agree when the measurements belong to different zones. Therefore, a correction might be necessary after the distillation of the private strings. Like [Linnartz and Tuyls 2003], we can see that the space is divided in exactly two interval, however the secret information is the quantization zone that the measurement belongs and the helper data tells us if the measurement is in the part odd or even of the quantization zone.

In the first version, the correction is done using BCH codes (described in [Bose and Ray-Chaudhuri 1960]) and transmitting the checksums bits. The leaked information is derived from the amount of communication needed to correct all the errors. In the second version, there is no direct correction, but the quantization steps are calculated in order to assure a low false negative rate.

One important point is that an adversary could try a man-in-the-middle attack. Every time that one party sends a bit b from the preliminary information, he sends $\neg b$ back. By doing this, the adversary determines how many positions he has to guess in order to discover the secret information. The solution proposed is to divided the information in two blocks and each party reveals one of them. Moreover, a minimum of samples with the same preliminary information (odd or even) in each partition is created. With this approach, impersonation becomes harder.

This work was an important advance in key conciliation with noisy information. However, the functions used discard too much information. The rate used for sampling is 220Hz and the rate of bits distilled for seconds is at maximum 20 bits, thus there is a great waste of information. A better result could be achieved if a better cryptography scheme were used.

[Mayrhofer and Gellersen 2007] and [Mayrhofer and Gellersen 2009] is another work that has created a key conciliation protocol based on acceleration. Differently from [Kirovski et al. 2007a] and [Kirovski et al. 2007b] that do the conciliation for each dimension, this work uses the magnitude over all normalized dimensions. After acquiring and processing the information, the protocol has two approaches to distill a key. The first

one uses Diffie-Hellman and Interlock to exchange the times series generated from each part. The security of this scheme is based on computational assumptions, which is undesirable. The second uses the Candidate Key Protocol described in [Mayrhofer 2007] to exchange the time series and extract a key. This approach is very dependent of how the processing of the feature was done, possibly a large amount of information is discarded and, also, high entropy is necessary to avoid off-line lookup table attacks. The lower bound of seven bits extracted per second is achieved in this work.

4.2. Authentication with Wireless Signal Power

[Premnath et al. 2014] created a protocol that measures the wireless signal power of a Bluetooth connection and, from it, derives a key between close devices. Some processing is necessary because of the low entropy of the measured power. Because of this, the bit rate distillation is considerably small, making this protocol less preferable in an real scenario.

4.3. Authentication with Biometry

In [Tuyls et al. 2005], another authentication system is built. The main focus is to use this in fingerprint recognition. A user has his feature measured multiple times and, for every measurement, if the value is greater than the global mean, then the output is 1, otherwise is 0. A feature is reliable if all the outputs are equal. The great disadvantages are the short keys and that there are few features that are really reliable.

5. Classes of codes compared

5.1. Reed-Solomon Codes

Reed-Solomon codes are the most well-known class of codes, appearing in CDs, NASA space missions, QR Codes, and many scientific papers, including [Schurmann and Sigg 2013]. For this reason they serve as a base line, and were therefore included in our comparison.

Reed Solomon codes are a subset of BCH codes and are linear block codes. A Reed-Solomon code is specified as $RS(n, k)$ with s-bit symbols. This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are $n - k$ parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where $2t = n - k$.

Reed-Solomon algebraic decoding procedures can correct errors and erasures. An erasure occurs when the position of an erred symbol is known. A decoder can correct up to t errors or up to $2t$ erasures. Erasure information can often be supplied by the demodulator in a digital communication system, i.e. the demodulator 'flags' received symbols that are likely to contain errors.

When a codeword is decoded, there are three possible outcomes:

1. If $2s + r < 2t$, where s is the number of erasures and r the number of erros, then the original transmitted code word will always be recovered.
2. The decoder will detect that it cannot recover the original code word and indicate this fact.

3. The decoder will mis-decode and recover an incorrect code word without any indication.

The probability of each of the three possibilities depends on the particular Reed-Solomon code and on the number and distribution of errors. For detailed information about encoding and decoding RS codes please refer to [Lin and Costello 2004].

5.2. Low-Density Parity Check Codes

Low-Density Parity Check Codes (LDPC) [Gallager 1962] are capacity achieving codes that works using the sum-product algorithm in the decoding process. The graph that this algorithm uses is known as the Tanner graph. It is a bipartite undirective graph. In one side we have the check nodes connected only to the variable nodes, that are on the other side. The check nodes take the value that is believed to be true and check if the constraint of the code is correct. Figure 1 from [Johnson 2009] shows an example of this graph.

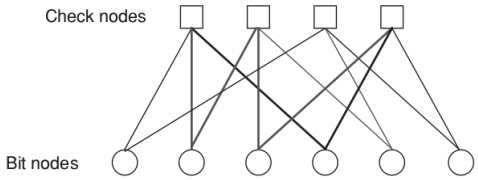


Figure 1. Tanner Graph of (1)

This graph represents the parity matrix:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad (1)$$

Each row correspond to a check node and each column to a variable node. The soft and hard decoding approaches can be used with the Tanner graph.

We have that the generator matrix G and parity matrix H have the property that $GH^t = 0$. The encoding is done simply by a matrix multiplication which can be optimized. Words c that belong to the code have the property that $Hc^t = 0$.

5.3. Repeat-Accumulate Codes

Repeat-Accumulate Codes are a simplification of LDPC codes. The encoding process is even simpler than the bipartite graphs of LDPC codes. The following example is taken from [MacKay 2003] (page 582): let the source bits be $s_1 \dots s_K$. Repeat each bit 3 (say) times: $s_1 s_1 s_1 \dots s_K s_K s_K$ and apply some arbitrary but fixed permutation on these bits, to obtain $u_1 u_2 u_3 \dots u_N$ with $N = 3K$. Transmit the accumulated sum: $t_1 = u_1$; $t_i = t_{i-1} \oplus u_i$ for $1 < i \leq N$.

RACs can also be represented through a Tanner graph and therefore decoding is also performed through the sum-product algorithm. RACs have a spectacular performance given their simplicity.

5.4. Low-Density Lattice Codes

Low-Density Lattice Codes (LDLC) [Sommer et al. 2008] are capacity achieving codes of the Unconstrained Power Channel (UPC) [Poltyrev 1994] and are based on the LDPC codes.

The UPC is simply an AWGN channel without the power restriction. So, because of this, another definition of capacity is used: instead of Signal Power, the volume of the lattice is used. This volume is defined as the generator matrix G of the code. In LDLC we have the restriction that $|det(G)| = 1$ and $G = H^{-1}$. In LDPC codes messages are encoded as bits and the distance notion is the Hamming distance. In LDLCs messages are represented as integer vectors with Euclidean distance; they can be interpreted as indexes; the encoded messages are lattice points, and decoding is performed using a sum-product algorithm adapted to the constraints.

One important aspect is that the UPC is merely a theoretical channel; it does not model a true communication scenario. In order to be applicable, the infinite region of the lattice has to be shaped into a finite one. See for example the approaches presented in [Sommer et al. 2009] and [Kurkoski et al. 2009].

Code-Offset Method with LDLC

Although in order to use the UPC in a real communication channel the shaping has to be done, in the code-offset method there is no actual communication, so there is no power restriction. Therefore, LDLC are suitable to be used directly. Compared to a channel like the BSC, we see that it can be used directly because in this specific case it is assumed that the quantization has already been done. By analyzing this, we can see that the shaping is done implicitly by the space of the feature itself.

6. Our simulations

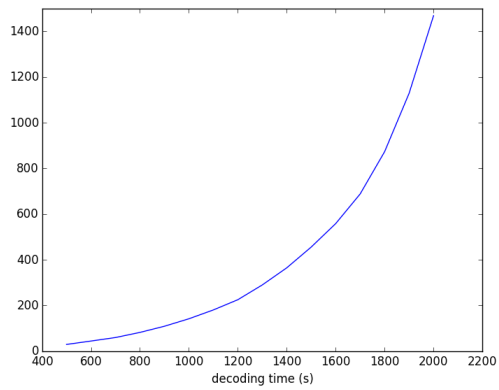
To compare these four classes of codes, we need a good implementation of each one. LDPC codes have a good library in python named pyldpc which we decided to use. We also found a good implementation of the Reed-Solomon codes in the wikiversity site. However we couldn't find implementations for the LDLC and Repeat-Accumulate codes, so we coded those ourselves.

Each Repeat-Accumulate Code is defined by some parameters, as explained before, so after we had a fully functional implementation it was also necessary to search for a good set of them. We did so by generating several random combinations and testing the resulting codes for decoding speed and correction capacity.

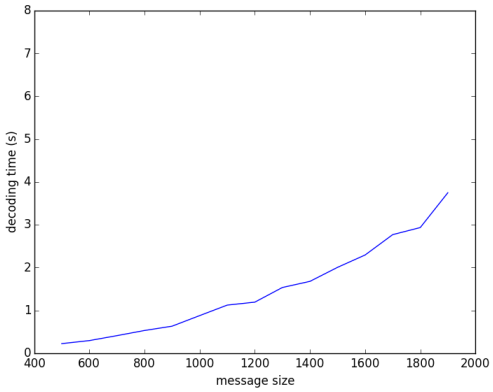
6.1. Comparing speed

To compare the decoding speed, we generated 5000 random messages with size varying from 500 to 2000 values (these values where bits in the LDPC and Repeat-Accumulate codes, and integers in LDLC and Reed-Solomon codes), we then added noise to them and proceeded with decoding.

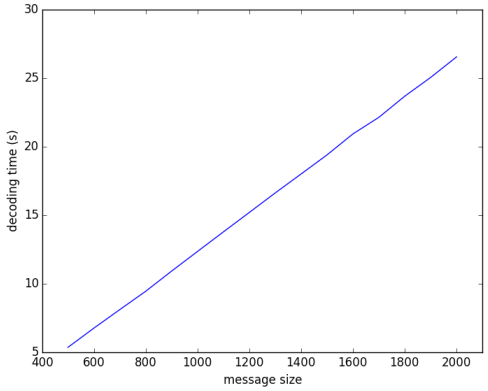
This process gave us the following results:



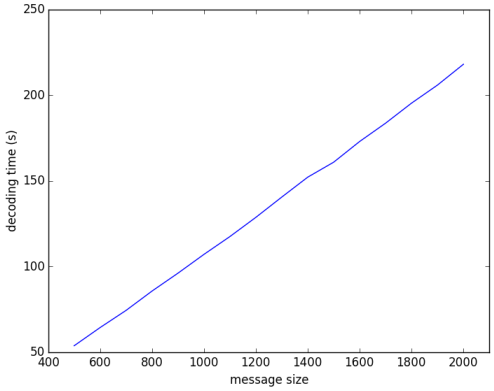
(a) Reed-Solomon Codes



(b) Low-Density Parity-Check Codes



(c) Repeat-Accumulate Codes



(d) Low-Density Lattice Codes

Figure 2. Message Size by Decoding Time

Observing these graphics we see that for small messages Reed-Solomon Codes are the fastest, but as the message grows its decoding time increases substantially.

Low-Density Lattice Codes' decoding time grows linearly with message size, however it is much slower than Low-Density Parity Check and Repeat-Accumulate codes, which also exhibit linear time complexity for decoding. This is a very interesting observation. When we first started working on this project we expected that LDLCs would be superior in every aspect. But practice showed us a complete different reality. Binary codes are much faster, and given the difference in magnitude and the smoothness of its growth we believe this to be a real advantage.

6.2. Comparing correction capability

To compare the codes according to correction capability, we again generate 5000 random messages of size varying from 500 to 2000, added noise, and then tried to correct them. One difference here is that we decided not to include Reed-Solomon Codes because their correction ability is known beforehand. We did not hperform simulations with Repeat-Accumulate Codes; it will be a future work to complete the comparisons adding these codes.

Low-Density Lattice Codes and Low-Density Parity-Check Codes both presented good results.

The noise we added was approaching both codes' limits, meaning that if it was a little higher, neither of them would be able to correct. But within the expected range they performed fine.

6.3. Comparing entropy

Another interesting criterion for comparing these codes is regarding their entropy, that is, how each one spreads its encoded messages in the coding space. That comparison was made only with LDPCs and LDLCs, since they were the most similar in construction (even if the theory behind is completely different) and the obvious difference of one being a binary code and the other a continuous one.

To do so, we treated the all zero message $\overrightarrow{0}$ as the codified message. We added noise to it and then tried to correct it, meaning we tried to correct noise alone, and observed how the codes behaved.

The process of adding noise, and trying to correct it was repeated around 5000 times using each code. After doing that we counted how many times each 'decoded message' appeared. Having these values, the minimum entropy was calculated according to:

$$H_{min} = \frac{count(m_{max})}{\sum_{i=0}^{n} count(m_i)} \tag{2}$$

Here $count(x)$ is the number of times message e was observed and n is the total number of different observed messages.

LDPC decoded all of his messages to the original one $\overrightarrow{0}$, which was to be expected given that the all zero message is a valid codeword for any LDPC instance, due to that its minimum entropy is 0!

LDLC gave us a more interesting result, we didn't get 2 equal messages, meaning that we found a total of 5000 different messages by correcting 5000 'noisy messages'. So according to the formula above, we got a total min entropy of 0.0002203 (we discarded messages that were not successfully corrected to valid code words).

7. Conclusions

With this work we concluded that LDPC codes' performance is still superior when it comes to speed, with Repeat-Accumulate Codes coming right after it.

Both LDPCs and LDLCs show great potential when it comes to correction capability, Reed-Solomon codes are the most trustworthy since its decoding capabilities are pre-defined and we can always make it as large as needed, Repeat-Accumulate codes are very similar to LDPCs but we still can't say much about its correcting capabilities, but due to its simplicity it is worth investigating in the future.

8. Acknowledgments

HdRD and JvdG were partially supported by grants from FAPEMIG (Universal 2013 and PROBIC). VPP was partially supported by the Brazilian agency CAPES. All three were partially supported by a grant from LG Electronics Brasil.

References

Bennett, C. H., Brassard, G., Crépeau, C., and Maurer, U. M. (1995). Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923.

Bose, R. C. and Ray-Chaudhuri, D. K. (1960). On a Class of Error Correcting Binary Group Codes. *Information and control*, 3(1):68–79.

Brassard, G. and Salvail, L. (1993). Secret-key reconciliation by public discussion. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 410–423. Springer.

Brassard, G. and Salvail, L. (1994). Secret-Key Reconciliation by Public Discussion. In *advances in Cryptology-EUROCRYPT 93*, pages 410–423. Springer.

Buhan, I., Doumen, J., Hartel, P., and Veldhuis, R. (2007a). Constructing Practical Fuzzy Extractors Using QIM. Technical report, Technical Report TR-CTIT-07-52, Centre for Telematics and Information Technology University of Twente.

Buhan, I., Doumen, J., Hartel, P., and Veldhuis, R. (2007b). Fuzzy Extractors for Continuous Distributions. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 353–355. ACM.

Chang, Y.-J., Zhang, W., and Chen, T. (2004). Biometrics-Based Cryptographic Key Generation. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 3, pages 2203–2206. IEEE.

Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. (2008). Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM journal on computing*, 38(1):97–139.

Gallager, R. (1962). Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28.

Goodrich, M. T., Sirivianos, M., Solis, J., Tsudik, G., and Uzun, E. (2006). Loud and Clear: Human-Verifiable Authentication Based on Audio. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 10–10. IEEE.

Johnson, S. J. (2009). *Iterative error correction: Turbo, low-density parity-check and repeat-accumulate codes*. Cambridge University Press.

Juels, A. and Sudan, M. (2006). A Fuzzy Vault Scheme. *Designs, Codes and Cryptography*, 38(2):237–257.

Kirovski, D., Sinclair, M., and Wilson, D. (2007a). The Martini Synch. Technical report, Technical Report MSR-TR-2007-123, Microsoft Research.

Kirovski, D., Sinclair, M., and Wilson, D. (2007b). The Martini Synch: Using Accelerometers for Device Pairing. Technical report, Citeseer.

Kumar, A., Saxena, N., Tsudik, G., and Uzun, E. (2009). Caveat emptor: A comparative study of secure device pairing methods. In *Seventh Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2009, 9-13 March 2009, Galveston, TX, USA*, pages 1–10.

Kurkoski, B. M., Dauwels, J., and Loeliger, H.-A. (2009). Power-constrained communications using ldlc lattices. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 739–743. IEEE.

Lin, S. and Costello, D. J. (2004). *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Linnartz, J.-P. and Tuyls, P. (2003). New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In *Audio-and Video-Based Biometric Person Authentication*, pages 393–402. Springer.

MacKay, D. J. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge university press.

Mayrhofer, R. (2007). The Candidate Key Protocol for Generating Secret Shared Keys from Similar Sensor Data Streams. In *Security and Privacy in Ad-hoc and Sensor Networks*, pages 1–15. Springer.

Mayrhofer, R. and Gellersen, H. (2007). Shake Well Before Use: Authentication Based on Accelerometer Data. In *Pervasive computing*, pages 144–161. Springer.

Mayrhofer, R. and Gellersen, H. (2009). Shake Well Before Use: Intuitive and Secure Pairing of Mobile devices. *Mobile Computing, IEEE Transactions on*, 8(6):792–806.

Poltyrev, G. (1994). On coding without restrictions for the awgn channel. *IEEE Transactions on Information Theory*, 40(2):409–417.

Premnath, S. N., Gowda, P. L., Kasera, S. K., Patwari, N., and Ricci, R. (2014). Secret Key Extraction Using Bluetooth Wireless Signal Strength Measurements. In *Sensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on*, pages 293–301. IEEE.

Reed, I. S. and Solomon, G. (1960). Polynomial Codes Over Certain Finite Fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304.

Schurmann, D. and Sigg, S. (2013). Secure Communication Based on Ambient Audio. *Mobile Computing, IEEE Transactions on*, 12(2):358–370.

Sommer, N., Feder, M., and Shalvi, O. (2008). Low-density lattice codes. *IEEE Transactions on Information Theory*, 54(4):1561–1585.

Sommer, N., Feder, M., and Shalvi, O. (2009). Shaping methods for low-density lattice codes. In *Information Theory Workshop, 2009. ITW 2009. IEEE*, pages 238–242. IEEE.

Tuyls, P., Akkermans, A. H., Kevenaar, T. A., Schrijen, G.-J., Bazen, A. M., and Veldhuis, R. N. (2005). Practical Biometric Authentication with Template Protection. In *Audio- and Video-Based Biometric Person Authentication*, pages 436–446. Springer.

Verbitskiy, E. A., Tuyls, P., Obi, C., Schoenmakers, B., and Skoric, B. (2010). Key Extraction from General Nondiscrete Signals. *Information Forensics and Security, IEEE Transactions on*, 5(2):269–279.

Zheng, G., Li, W., and Zhan, C. (2006). Cryptographic Key Generation from Biometric Data Using Lattice Mapping. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 513–516. IEEE.

