University of Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Quantization for Secret Key Generation in Underwater Acoustic Channels

Supervisor

Professor Stefano Tomasin

University of Padova

Co-supervisor

Francesco Ardizzon

University of Padova

Master Candidate

Amedeo Giuliani

Academic Year

2021-2022

To Gloria, who has always motivated and
supported me, especially during hard times.

Abstract

Securing wireless communications in harsh environments, such as underwater networks, via traditional cryptographic approaches is unfeasible. For example, public key encryption would require a public key infrastructure and a key management infrastructure. A viable solution is instead physical layer security, allowing two devices to obtain a symmetric cryptographic key from the randomness provided by the underlying communication channel, which varies in time, frequency, and space, in general.

The probability of having both parties generating the same key and its number of bits greatly depend on how sampled observations are quantized. In this thesis, novel data-driven quantization techniques, which make use of specific channel features computed from impulse responses collected from real experiments, are investigated. In particular, we propose a new machine learning algorithm that quantizes an input vector into an initial key, as close as possible to a series of independent and uniformly distributed symbols and matches at beast the corresponding initial key of the corresponding receiver, to guarantee a high key agreement probability and to avoid an eavesdropper to infer future values exploiting the correlation between consecutive symbols. We also propose an adversarial neural network architecture, where legitimate parties feature a neural quantizer to produce the initial key, whereas the eavesdropper tries to reconstruct the key agreed by the first two.

Contents

Listing of figures

Listing of tables

Listing of acronyms

UWA . . . . . Underwater Acoustic

UAN . . . . . Underwater Acoustic Network

ICI . . . . . Inter-Carrier Interference

ISI . . . . . Inter-Symbol Interference

CIR . . . . . Channel Impulse Response

CFR . . . . . Channel Frequency Response

SKA . . . . . Secret Key Agreement

SKG . . . . . Secret Key Generation

AWGN . . . . . Additive White Gaussian Noise

PLS . . . . . Physical Layer Security

TDD . . . . . Time Division Multiplexing

RSS . . . . . Received Signal Strength

SL . . . . . Spreading Loss

AL . . . . . Absorption Loss

PSD . . . . . Power Spectral Density

IEEE . . . . . Institute of Electrical and Electronics Engineers

ML . . . . . Machine Learning

NL . . . . . Neural Network

ReLU . . . . . Rectified Linear Unit

MSE . . . . . Mean Square Error

PCA . . . . . Principal Component Analysis

t-SNE . . . . . t-distributed stochastic neighbor embedding

MTAE . . . . . Multi-Task Autoencoder

DAAE . . . . . Domain-Adversarial Training of Autoencoder

GRL . . . . . Gradient Reversal Layer

KGnet . . . . . Key Generation neural network

IPI . . . . . Inter-Pulse Interval

WBAN . . . . . Wireless Body Area Network

PMD . . . . . Probability Mass Distribution

RMS . . . . . Root Mean Square

KDE . . . . . Kernel Density Estimation

KAP . . . . . Key Agreement Probability

SKR . . . . . Secret Key Rate

SVM . . . . . Support Vector Machine

SMR . . . . . Symbol Matching Rate

DSQ . . . . . Differentiable Soft Quantization

AAE . . . . . Adversarial Autoencoder

1 Introduction

Today's world depends heavily on wireless communications, and data transfer through these networks is growing for a variety of cases, including civilian applications like social networking, banking, and other financial and business operations, but more importantly for military applications. Information security has become a primary concern as a result of the exponential increase of wireless services, as individuals rely on the wireless network to send sensitive data. Due to the broadcast nature of wireless communication systems, any user within range can intercept signals being exchanged between two parties, giving a possible attacker the opportunity to initiate a series of passive attacks such as eavesdropping, or to carry out active attacks like denial of service, spoofing, and jamming. Therefore, there is a strong need to secure the transmitted information.

Traditional security approaches, namely, encryption-based schemes in which the message is encrypted via some algorithm, rely on the fact that such algorithm is complex enough to prevent a possible attacker from decrypting the intercepted message. However, with the recent advantages in computational techniques, there is a possibility for the attacker to steal the key via brute force attacks. On the contrary, physical layer security schemes do not depend on computational complexity but they exploit unpredictable and random characteristics of wireless channels, e.g., noise and fading. Specifically, legitimate users independently generate a symmetric key from the randomness of the underlying wireless channel.

The focus of this thesis work is physical layer security applied to underwater acoustic channels, with a focus on quantization methods with data-driven approaches, namely, machine

learning techniques and neural network architectures.

The thesis is structured as follows.

Chapter 2 starts with an overview of the UWA channel, explaining the characteristics of the medium and how to model the path loss and the noise in underwater communications.

Chapter 3 introduces the core concepts of PLS, describes the steps of the general SKA procedure, and terminates with a review of the state of the art.

Chapter 4 describes an initial version of our quantizer, exploring several possible loss functions and relative outcomes. Then, it is seen that the presence of an eavesdropper observing a highly correlated channel with respect to one of the legitimate parties leads to a low secret key rate if nothing is done to prevent this. Ultimately, the two-dimensional initial version is extended to accept in input an arbitrary number of channel features.

In Chapter 5 the attention is shifted towards neural networks and in particular on developing a neural quantizer that is capable of producing a key that is consistent for the legitimate parties, but not for the eavesdropper.

Chapter 6 gathers the numerical results of every algorithm and neural network architecture developed in the thesis.

2

Underwater Acoustic Communication

2.1 Underwater Acoustic channel

In the *Underwater Acoustic* (UWA) channel communication does not occur with radio waves due to the high rate of absorption of electromagnetic signals, but acoustic waves are employed, which have a much lower propagation speed, namely $v \simeq 1.5 \cdot 10^3$ m/s and because of this, latency is quite high, namely, ~ 0.67 ms/m. An acoustic modem usually operates at frequencies ranging from $\sim$ 10 kHz to $\sim$ 100 kHz. Due to limited bandwidth and high end-to-end delay, transmission rate is of the order of few tens of kbps. Moreover, although for the mobile radio channel lots of models have been developed, it does not exist a standard model for UWA communication channels and hence one has to carry out experimental analyses to understand the statistical properties of the UWA channel at the particular site the system has to be installed.

2.1.1 Channel modeling

The UWA channel is a doubly selective channel, meaning large Doppler and delay spread. They arises because of time-varying characteristics of seawater and the surrounding environment. Specifically, the Doppler spread is mainly given by the change of path length inducted by surfaces waves and relative motion of transmitter or receiver, whereas the delay spread is due to the presence of several delayed copies of the signal arriving at the receiver because of reflections at sea surface and bottom and any other object present, and refraction in water. The first results

	Shallow waters	Deep waters	Practical
Spreading loss model	cylindrical wave	spherical wave	cylindrical/spherical wave
Spreading loss factor	$k = 1$	$k = 2$	$k = 1.5$

Table 2.1: Spreading loss model [6].

in a time-selective behavior and causes *Inter-Carrier Interference* (ICI), while the second results in a frequency-selective behavior and is accountable for *Inter-Symbol Interference* (ISI).

Transmission in a UWA channel is also strongly affected by path loss, which is composed by *spreading loss* (SL) and *absorption loss* (AL). SL is caused by the "spreading" of the acoustic wave, i.e., as the wave travels towards the receiver its front will occupy a larger and larger area, which means that the energy per surface unit becomes less and less. AL is caused by the energy of the acoustic wave being converted in other forms – such as heat, which is then absorbed by the medium – as it travels towards the receiver. The overall path loss over a distance l for a signal at frequency f is given by [7]:

$$A(l, f) = A_0 l^k a(f)^l \tag{2.1}$$

where $a(f)$ is the absorption coefficient, which varies in frequency and can be expressed via Thorp's formula for frequencies less than 50 kHz [8]:

$$10 \log a(f) = \frac{0.11 f^2}{1 + f^2} + \frac{44 f^2}{4100 + f^2} + 2.75 \cdot 10^{-4} f^2 + 0.003 \quad \text{dB/km} \tag{2.2}$$

or via Fisher and Simmons' formula, that also takes into account the depth d and temperature t [9]:

$$10 \log a(d, t, f) = \frac{A_1 P_1 f_1 f^2}{f_1^2 + f^2} + \frac{A_2 P_2 f_2 f^2}{f_2^2 + f^2} + A_3 P_3 f^2 \quad \text{dB/km} \tag{2.3}$$

In Table 2.1 are reported the possible values for the spreading factor k in (2.1). In shallow waters the transmission distance is less than the height of water, while in deep waters it holds the converse. Usually, the spreading factor is set to $k = 1.5$ for practical applications.

As in any other environment, there is also noise hindering communications. In particular, noise in underwater channels can be classified into two groups: *ambient noise*, always present in background and caused by environmental conditions and objects moving in the sea, and *site-specific noise*, which occurs only in precises locations such as polar regions. Regarding ambient noise, there are many sources which are more or less influent depending on the operating frequency:

- turbulence noise, which is dominant for frequencies $f < 10$ Hz
- shipping noise, that influences mostly the interval $10 < f < 100$ Hz
- waves and other surface motions caused by rains and winds, which impacts mainly the interval $100 < f < 100$ kHz
- thermal noise, that becomes dominant for frequencies $f > 100$ kHz

The *Power Spectral Density* (PSD) for each of these four sources is empirically modeled as follows:

$$\begin{aligned} 10 \log N_t(f) &= 17 - 30 \log f \\ 10 \log N_s(f) &= 40 + 20(s - 0.5) + 26 \log f - 60 \log(f + 0.03) \\ 10 \log N_w(f) &= 50 + 7.5\sqrt{w} + 20 \log f - 40 \log(f + 0.4) \\ 10 \log N_t h(f) &= -15 + 20 \log f \end{aligned} \tag{2.4}$$

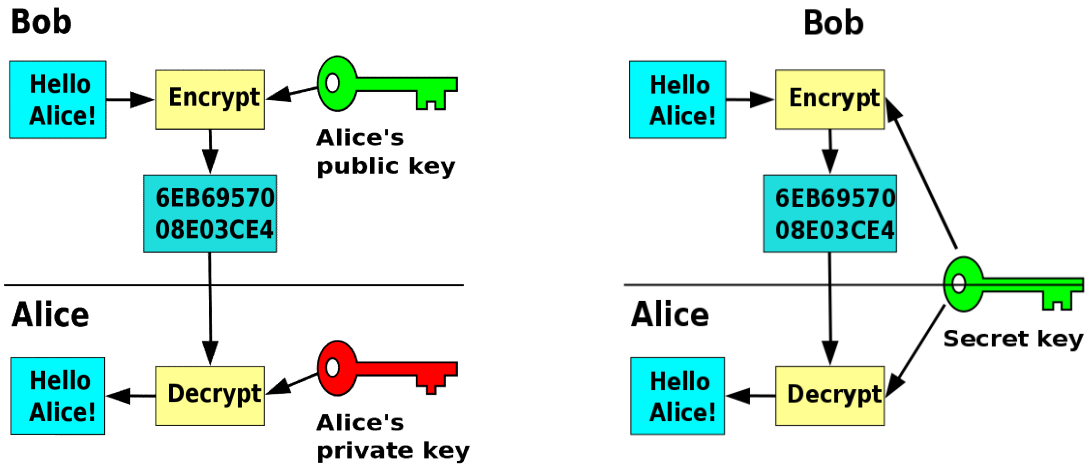
where $s \in [0, 1]$ indicates the intensity of the shipping activity and w is the wind speed in m/s [7][6][10][11].

3

Secret Key Generation

3.1 Physical Layer Security

Since *Underwater Acoustic Networks* (UANs) are based on acoustic communication, which means they use a broadcast wireless channel, an attacker could easily eavesdrop the signals travelling over the medium. Hence, it is necessary for the nodes in the network to implement secure ways to communicate. Classical encryption schemes (Fig. 3.1) are the symmetric encryption, where the nodes use the same key to encrypt and decrypt data, and the use of asymmetric encryption, in which each user has a pair of keys, public and private. Whenever a node wants to transmit something to another device, it will use the public key of that device to encrypt the data; the receiving node will then decrypt the received data with its private key. However, in the first case one would have to implement a key distribution center, which is almost an impossible task in UANs. A possible solution could be generating keys via a pseudorandom generator and pre-installing them in the nodes, but the lack of randomness of such generators could be exploited and when a certain key is compromised, all nodes using that key are not secure anymore. Instead, in the second case, there is the need of a dedicated infrastructure for key management and authentication. Moreover, a public key encryption scheme requires high computing power since they are based on mathematical problems. A more promising approach for UANs is the *Physical Layer Security* (PLS), in which the aim is to dynamically generate secret keys exploiting the randomness and reciprocity of the common acoustic channel, the first



(a) Public key scheme.

(b) Symmetric key scheme.

Figure 3.1: Symmetric and asymmetric encryption schemes visual comparison.

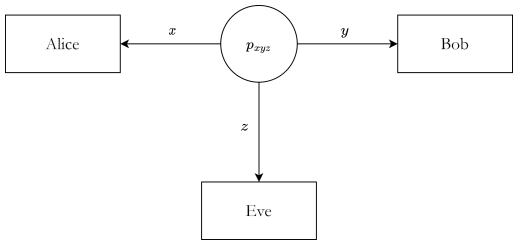
given by time-varying propagation physics in UWA channels. This paradigm is not computationally demanding and it is information-theoretically secure, making it a good solution for UANs, in which nodes are usually power constrained.

Throughout the whole thesis, we will refer to Alice and Bob as the two legitimate parties that want to distill a secret symmetric key to use for subsequent secure communications, while we will refer to Eve as the eavesdropper, i.e., a malicious node that is only capable of passive attacks, the aim of which is to steal the key used by Alice and Bob.

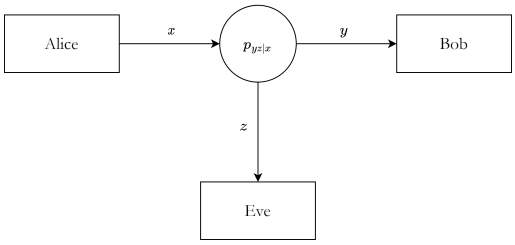
3.1.1 Source and channel models

In the **source model**, shown in Fig. 3.2a, Alice, Bob, and Eve respectively observe the statistically related [noisy] observations x, y, and z of a common source of randomness – which is the channel itself – modeled as a memory-less source $(\mathcal{X}\mathcal{Y}\mathcal{Z}, p_{xyz})$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are the sets in which x, y, z take their values respectively, and p_{xyz} is a collection of probability mass functions, one for each x, y, z . It is assumed that the channel statistics are know but it cannot be controlled by the involved parties by any means. This is the model adopted in the thesis.

In the **channel model**, shown in Fig. 3.2b, instead of observing realizations of an external source, Bob and Eve observe the outputs of the channel, modeled as a memory-less channel $(\mathcal{X}, p_{yz|x}, \mathcal{Y}\mathcal{Z})$, where the input set $\mathcal{X}$ is controlled by Alice and $p_{yz|x}$ is a collection of probability mass functions, one for each $\mathcal{X}$. In this case the source of randomness is hence Alice



(a) Source-type.



(b) Channel-type.

Figure 3.2: Source and channel models in the PLS approach.

itself [12].

3.1.2 Secret Key Agreement protocol

The general *Secret Key Agreement* (SKA) protocol, also known as *Secret Key Generation* (SKG) procedure, is visualized in Fig. 3.3 and also summarized as follows [13]:

1. Channel probing
   (a) Channel sampling
   (b) Pre-processing
2. Quantization
3. Information reconciliation
4. Privacy amplification

The channel probing step further divides into channel sampling, in which Alice and Bob exchange probing signals to estimate either the *Channel Impulse Response* (CIR) or the *Channel Frequency Response* (CFR), and pre-processing, in which signal processing techniques are implemented to manipulate the acquired data before proceeding with subsequent steps. Then, there is the quantization step, where the measurements are translated into bits, yielding the initial key. Since Alice and Bob exchange probe signals in a *Time Division Duplexing* (TDD) fashion, and due to hardware differences of receivers, the two quantized sequences will likely not be equal, even though the probing signals are exchanged within the channel coherence time. Hence, in the information reconciliation stage the aim is to correct the mismatching bits in Alice and Bob initial keys. However, this operation leaks some information on the public

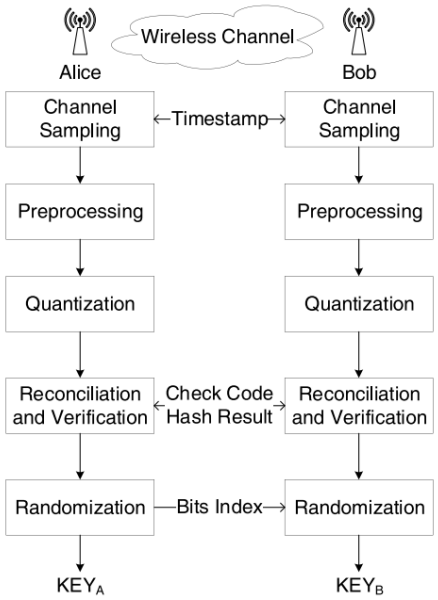


Figure 3.3: SKA protocol [1].

channel that Eve could exploit. Thus, as last step, privacy amplification has the goal to distill a shorter and more secure key through, e.g., a pre-agreed hash function.

3.1.3 Secret key capacity for source model

The weak secret key capacity $\mathrm{C_s^{SM}}$ for the source model described in Section 3.1.2 is the maximum number of bits that Alice and Bob can obtain from the respective channel observation that are unknown to Eve, and can be bounded as [12]:

$$\mathrm{I}(X;Y) - \min\{\mathrm{I}(X;Z), \mathrm{I}(Y;Z)\} \leq \mathrm{C_s^{SM}} \leq \min\{\mathrm{I}(X;Y), \mathrm{I}(X;Y \mid Z)\} \tag{3.1}$$

The lower bound is simply given by the difference between the information rate between the legitimate parties, Alice and Bob, and a certain amount of information leaked to the eavesdropper Eve. In this thesis we will maximize $\mathrm{C_s^{SM}}$ by maximizing this lower bound.

3.2 SKA protocol state of the art

Most existing implementations of the SKA protocol are based on classical signal processing, and they are mostly for *Institute of Electrical and Electronics Engineers* (IEEE) 802.11 and IEEE

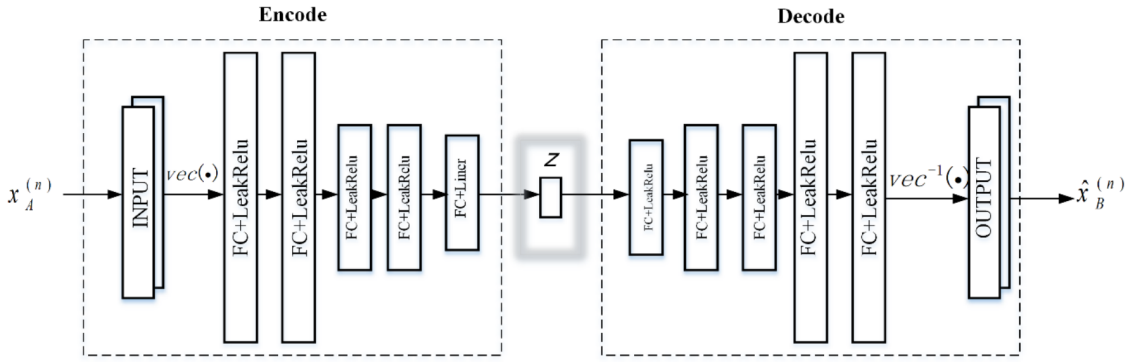


Figure 3.4: AE architecture [2].

802.15.4. Moreover, popular observations to extract the key from are CIR, CFR, and the *Received Signal Strength* (RSS) [13], but there is also the possibility to distill a key from other features.

In this thesis we focus on data-driven approaches, thus we will review some works employing *Machine Learning* (ML) algorithms and *Neural Network* (NN) architectures.

3.2.1 Neural Network implementations

The authors of [2] propose an SKG scheme based on an autoencoder (AE) to extract reciprocal features from weakly correlated CIR estimates of radio channels. The structure of the proposed AE is shown in Fig. 3.4. The inputs $\boldsymbol{x}_A^{(i)}$ are Alice's CIR estimates $\hat{\boldsymbol{h}}_A^{(i)}$, of size 300×2, each row of which is a pair of amplitude and phase of the corresponding tap, i.e.,

$$\boldsymbol{x}_u^{(i)} = [\mathbf{A}_u^{(i)}, \boldsymbol{\varphi}_u^{(i)}], \tag{3.2}$$

where $\mathbf{A}_u^{(i)}$ and $\boldsymbol{\varphi}_u^{(i)}$ are the amplitude and phase of each user CIR $\hat{\boldsymbol{h}}_u^{(i)}$, respectively. The inputs are first reshaped into a one-column vector, and then they are passed through the encoder $f_{enc}(\cdot;\theta)$ – with θ the NN learnable parameters – composed by five fully-connected layers of size $1000, 1000, 500, 500$, and 300, respectively. The first four layers have the Leaky *Rectified Linear Unit* (ReLU) as activation function, while for the last one a linear activation function is used to output the encoded vector $z^{(i)}$ of size 10. This vector is passed to the decoder $g_{dec}(\cdot;\theta')$ – composed by five fully-connected layers of size $300, 500, 500, 1000$, and 1000, respectively, all of them equipped with a Leaky ReLU activation function – to output $\boldsymbol{x}_B^{(i)}$, which should be as close as possible to $\boldsymbol{h}_B^{(i)}$.

Since the number of scatters in a radio channel is limited, the channel responses are sparse in terms of power or amplitude. Hence, the phase of channel responses with low power is worthless to be recovered and counted in the loss function, and so this has been redefined to emphasize the influence of large amplitude part:

$$\ell^{(i)} = \left\| \mathbf{A}_B^{(i)} - \hat{\mathbf{A}}_B^{(i)} \right\|_2 + \rho \sum_j \left(\mathbf{A}_B^{(i,j)} \right)^2 \left\| \boldsymbol{\varphi}_B^{(i,j)} - \hat{\boldsymbol{\varphi}}_B^{(i,j)} \right\|_2^2 \tag{3.3}$$

where $\mathbf{A}_B^{(i)}$ and $\boldsymbol{\varphi}_B^{(i)}$ represent the amplitude and phase of target output $\boldsymbol{h}_B^{(i)}$, while $\hat{\mathbf{A}}_B^{(i)}$ and $\hat{\boldsymbol{\varphi}}_B^{(i)}$ are the amplitude and phase of the output of the decoder $\boldsymbol{x}_B^{(i)}$. Instead, the weight parameter ρ controls the proportion of the amplitude and phase error.

Note that since input and output do not coincide, the proposed AE works as a denoising AE. As long as the difference between the output and Bob's estimates becomes small, the extracted features $\boldsymbol{z}$ may represent the common main part of Alice and Bob's side-information.

The optimal encoding and decoding parameters θ^* and θ'^* can be obtained when the average reconstruction error (namely, the average over the training set of (3.3)) reaches global minimum. After the AE training, Alice and Bob can generate reciprocal features as:

$$\begin{aligned} \boldsymbol{z}_A &= f_{enc}(\hat{\boldsymbol{h}}_A^{(i)}; \theta^*), \\ \boldsymbol{z}_B &= f_{enc}(\hat{\boldsymbol{h}}_B^{(i)}; \theta^*). \end{aligned} \tag{3.4}$$

To assess the performance of this architecture, the authors have chosen the following two metrics:

1. the *Mean Square Error* (MSE) between the extracted reciprocal features of Alice and Bob

$$\mathrm{MSE} = \frac{1}{m} \sum_{i=1}^{m} \left\| \boldsymbol{z}_A^{(i)} - \boldsymbol{z}_B^{(i)} \right\|_2^2, \tag{3.5}$$

   where m is the number of samples in the test set;

2. the mutual information between the extracted reciprocal features of Alice and Bob

$$\mathrm{MI} = \mathrm{I}(\boldsymbol{z}_A, \boldsymbol{z}_B) = \sum_{\boldsymbol{z}_a} \sum_{\boldsymbol{z}_b} p(\boldsymbol{z}_a, \boldsymbol{z}_b) \log \frac{p(\boldsymbol{z}_a, \boldsymbol{z}_b)}{p(\boldsymbol{z}_a) p(\boldsymbol{z}_b)}, \tag{3.6}$$

   where p denotes the probability mass function.

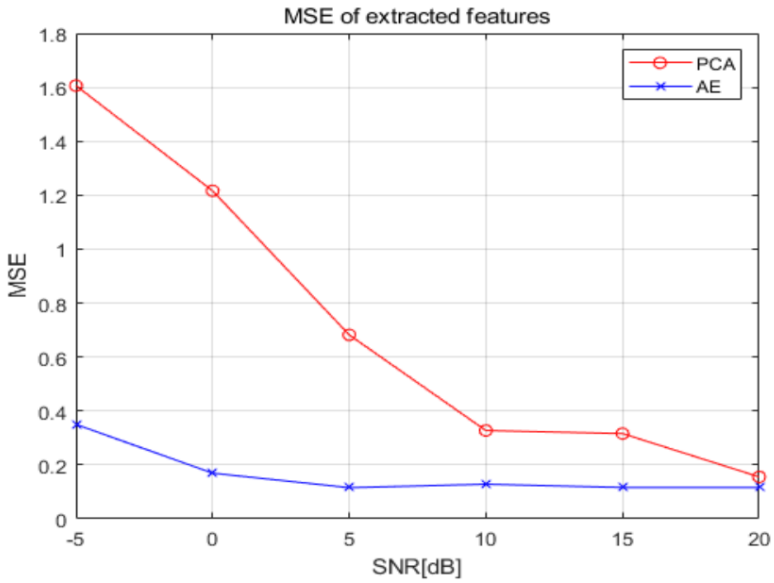


Figure 3.5: The MSE of $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ based on PCA and AE [2].

The MSE and MI results of the AE solution are compared with the ones obtained with *Principal Component Analysis* (PCA). As shown in Fig. 3.5 and Fig. 3.6, the proposed scheme always outperforms the PCA algorithm, and this holds especially for low values of SNR, thanks to the additional degrees of freedom of non-linear transformations.

This work assumes that an eavesdropper (Eve) is far from legitimate sides by at least $\frac{\lambda}{2}$, i.e., spatial decorrelation occurs, and the correlation between Eve's CIR estimates and Alice and Bob estimates is so weak that can be ignored. However, when Eve is at a distance less than half a wavelength the correlation between Eve and Alice's CIR estimates is significant, allowing Eve to steal keys using its own CIR estimates with the same network as the legitimate sides. The authors of [3] propose a *Multi-Task Autoencoder* (MTAE), that extracts the reciprocal features from the channel estimates of the legitimate sides while reducing its correlation with the eavesdropper. The feature extractor $G_z(\cdot;\theta_z)$ tries to extract the main reciprocal information between legitimate communication sides, while attempting to make this information irrelevant for Eve. In this way, even if the eavesdropper manages to obtain the same network that Alice and Bob use for reciprocal features extraction, it will be difficult to obtain the same key by using its features. In particular, the inputs $\boldsymbol{x}_{\mathrm{A}}^{(i)}, \boldsymbol{x}_{\mathrm{E}}^{(i)} \in \mathbb{R}^{300\times 2}$ are first reshaped into a one-column vector and then mapped into D-dimensional feature vectors $\boldsymbol{z}_{\mathrm{A}}^{(i)}$ and $\boldsymbol{z}_{\mathrm{E}}^{(i)}$ by $G_z(\cdot;\theta_z)$, which consists in four fully-connected layers of size $600, 300, 120$, and 64, respectively. The

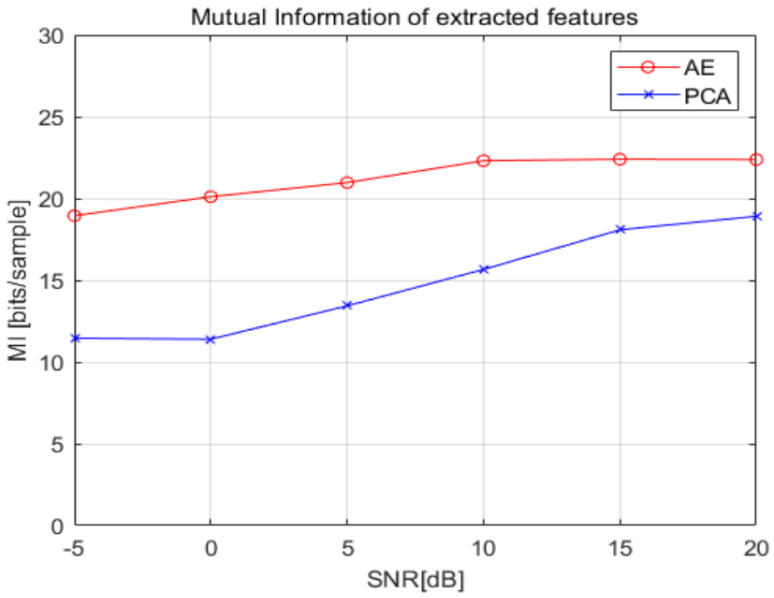


Figure 3.6: The MI of $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ based on PCA and AE [2].

first three layers use a Leaky ReLU, while the last one has a linear activation function. Then, $\boldsymbol{z}_{\mathrm{A}}^{(i)}$ is mapped to $\hat{\boldsymbol{x}}_{\mathrm{B}}^{(i)}$ – which should be as close as possible to $\boldsymbol{x}_{\mathrm{B}}^{(i)}$ – by decoder $G_b(\cdot;\theta_b)$, that is the converse of the feature extractor and consists of four fully-connected layers all equipped with a Leaky ReLU. The MTAE architecture is visualized in Fig. 3.7.

The proposed loss function for the i-th sample is:

$$\begin{aligned}\ell^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)},\boldsymbol{x}_{\mathrm{B}}^{(i)},\boldsymbol{x}_{\mathrm{E}}^{(i)};\theta_z,\theta_b) &= \ell_b^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)},\boldsymbol{x}_{\mathrm{B}}^{(i)};\theta_z,\theta_b) - \lambda\,\ell_z^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)},\boldsymbol{x}_{\mathrm{E}}^{(i)};\theta_z,\theta_b) = \\ &= \left\|\boldsymbol{x}_{\mathrm{B}}^{(i)} - \hat{\boldsymbol{x}}_{\mathrm{B}}^{(i)}\right\|_2^2 - \lambda\left\|\boldsymbol{z}_{\mathrm{A}}^{(i)} - \boldsymbol{z}_{\mathrm{E}}^{(i)}\right\|_2^2 = \\ &= \left\|\boldsymbol{x}_{\mathrm{B}}^{(i)} - G_b(G_z(\boldsymbol{x}_{\mathrm{A}}^{(i)};\theta_z);\theta_b)\right\|_2^2 - \\ &- \lambda\left\|G_z(\boldsymbol{x}_{\mathrm{A}}^{(i)};\theta_z) - G_z(\boldsymbol{x}_{\mathrm{E}}^{(i)};\theta_z)\right\|_2^2,\end{aligned} \tag{3.7}$$

where the parameter λ, which controls the trade-off between the two objectives of the MTAE, is adjusted dynamically during the training with the following schedule:

$$\lambda = \lambda_0\left(1 - \frac{1}{1+e^{-3\frac{epoch}{\mathrm{EPOCH}}}}\right) \tag{3.8}$$

where λ_0 is the parameter starting value, $epoch$ is the current number of epochs elapsed, and

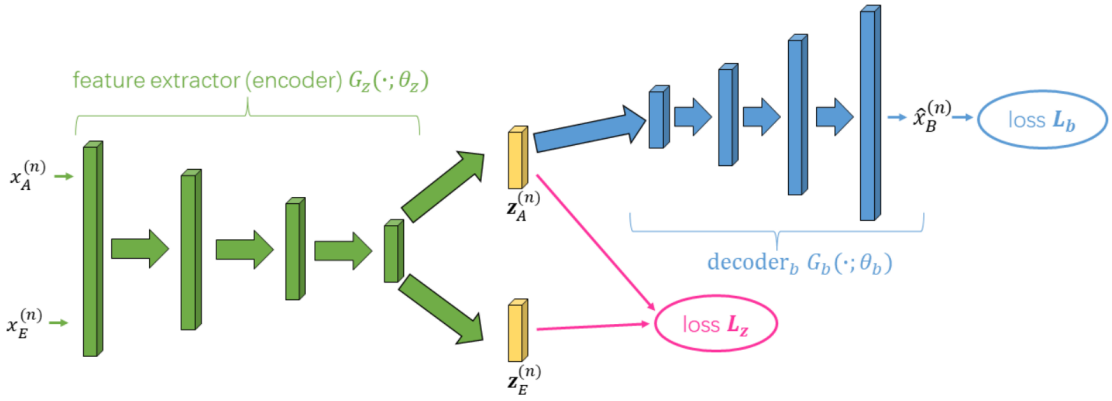


Figure 3.7: MTAE architecture [3].

EPOCH is the total number of epochs required for the training.

The overall loss function is:

$$\mathcal{L}_{\text{MTAE}}(\mathcal{S};\theta_z,\theta_b) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \ell_b^{(i)}(\boldsymbol{x}_{\text{A}}^{(i)},\boldsymbol{x}_{\text{B}}^{(i)};\theta_z,\theta_b) - \lambda\,\ell_z^{(i)}(\boldsymbol{x}_{\text{A}}^{(i)},\boldsymbol{x}_{\text{E}}^{(i)};\theta_z,\theta_b) \tag{3.9}$$

where $\mathcal{S}$ is the training set. Learning occurs by seeking the optimum weights θ_z^*, θ_b^* – for which the saddle point of (3.9) is achieved – by minimizing (3.9):

$$(\theta_z^*,\theta_b^*) = \arg\min_{\theta_z,\theta_b} \mathcal{L}_{\text{MTAE}}(\mathcal{S};\theta_z,\theta_b). \tag{3.10}$$

The trained feature extractor is equipped both on Alice and Bob to generate the reciprocal features as:

$$\begin{aligned} \boldsymbol{z}_A &= G_z(\boldsymbol{x}_{\text{A}}^{(i)};\theta_z^*) \\ \boldsymbol{z}_B &= G_z(\boldsymbol{x}_{\text{A}}^{(i)};\theta_z^*) \end{aligned} \tag{3.11}$$

To understand how good is this architecture at reducing the possibility of Eve stealing the key of Alice and Bob, the authors choose the following two metrics:

1. the Pearson correlation coefficient

$$\rho = \frac{COV(\boldsymbol{z}_{u_1},\boldsymbol{z}_{u_2})}{\sigma_{\boldsymbol{z}_{u_1}},\sigma_{\boldsymbol{z}_{u_2}}} \tag{3.12}$$

where $u_1, u_2 \in \text{A, B, E}$ and $u_1 \neq u_2$, to see by how much the correlation between Alice and Bob is reduced;

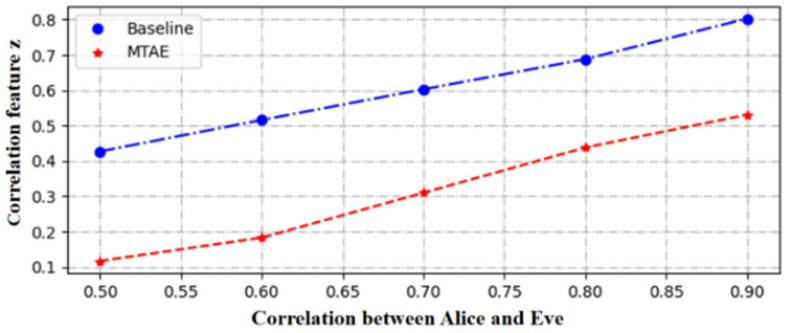


Figure 3.8: Correlation coefficient of the feature z between Alice and Eve [3].

Figure 3.9: t-SNE visualization of features [3].

2. *t-distributed stochastic neighbor embedding* (t-SNE) visualization, a dimensionality reduction technique to visualize the latent space z. If two data are similar in a high-dimensional space, they should be close to each other when reduced to a 2-dimensional space, and vice versa.

These two metrics are also computed for the baseline architecture, namely, the AE model in [2] for a comparison.

In Fig. 3.8 it can be clearly seen that the features generated with MTAE have a lower correlation than those generated through the naive AE. Also from Fig. 3.9 is obvious that the MTAE increases the differentiation between z_{A} (yellow dots) and z_{E} (purple dots).

Another approach to ensure information security when the eavesdropper is close less than half a wavelength from either legitimate parties is explained in [4], in which the same authors propose a *Domain-Adversarial Training of Autoencoder* (DAAE). The main components of such architecture are: one encoder, two decoders, and a *Gradient Reversal Layer* (GRL). As

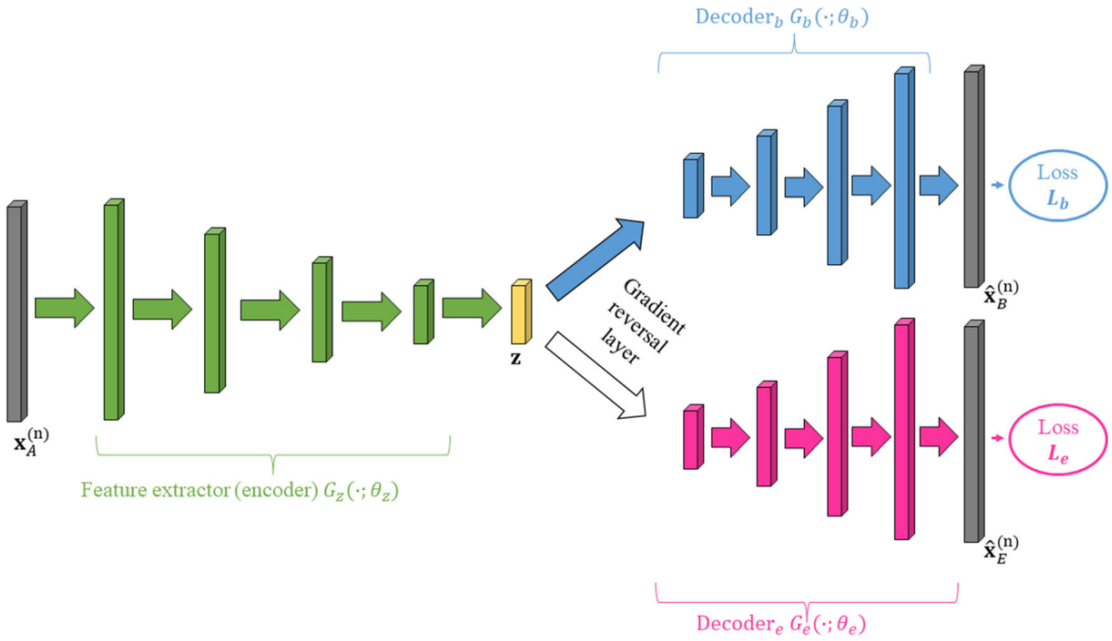


Figure 3.10: DAAE architecture [4].

for the MTAE, the goal is to extract the reciprocal channel features of legitimate sides while maximizing the feature difference with respect to Eve. Specifically, the input Alice's CIR $\boldsymbol{x}_{\mathrm{A}}^{(i)} \in \mathbb{R}^{300\times 2}$ is reshaped into a one-column vector and it is mapped to the D-dimensional vector $\boldsymbol{z}$ by the feature extractor $G_z(\cdot;\theta_z)$. Then, the feature vector is mapped to $\hat{\boldsymbol{x}}_{\mathrm{B}}^{(i)}$ – which again should be as close as possible to Bob's estimate $\boldsymbol{x}_{\mathrm{B}}^{(i)}$ – by decoder $G_b(\cdot;\theta_b)$. Ultimately, $\boldsymbol{z}$ goes through the GRL before being mapped to $\hat{\boldsymbol{x}}_{\mathrm{E}}^{(i)}$ – which should be as far as possible to Eve's estimate $\boldsymbol{x}_{\mathrm{E}}^{(i)}$ – by decoder $G_e(\cdot;\theta_e)$. The feature extractor consists in four fully-connected layers of size $600, 300, 120$, and 64, respectively, all of them equipped with a Leaky ReLU activation function but the last, which has a linear activation function. The two decoders have the same architecture, four fully-connected layers of size $64, 120, 300$, and 600, respectively, all using the LeakyReLU as activation function. The overall architecture is illustrated in Fig. 3.10. In particular, the proposed training method has a two-fold objective:

1. minimize the reconstruction error between the outputs $\hat{\boldsymbol{x}}_{\mathrm{B}}^{(i)}$ and the ground truth $\boldsymbol{x}_{\mathrm{B}}^{(i)}$ to let the encoded vector $\boldsymbol{z}$ containing the reciprocal information between Alice and Bob's CIR estimates;

2. maximize the reconstruction error between the outputs $\hat{\boldsymbol{x}}_{\mathrm{E}}^{(i)}$ and the ground truth $\boldsymbol{x}_{\mathrm{E}}^{(i)}$ to ensure that $\boldsymbol{z}$ contains as little information as possible related to Eve's CIR estimates.

The first objective can be achieved with a standard AE, and thus the networks G_z and G_b are trained by descending the gradient of the following loss function:

$$\ell_b^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)}, \boldsymbol{x}_{\mathrm{B}}^{(i)}; \theta_z, \theta_b) = \left\| \boldsymbol{x}_{\mathrm{B}}^{(i)} - G_b(G_z(\boldsymbol{x}_{\mathrm{A}}^{(i)}; \theta_z); \theta_b) \right\|_2^2. \tag{3.13}$$

The second objective instead can be achieved by seeking the converse objective of the AE, that is, the networks G_z and G_e are trained by ascending the gradient of the loss function:

$$\ell_e^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)}, \boldsymbol{x}_{\mathrm{E}}^{(i)}; \theta_z, \theta_e) = \left\| \boldsymbol{x}_{\mathrm{E}}^{(i)} - G_b(G_z(\boldsymbol{x}_{\mathrm{A}}^{(i)}; \theta_z); \theta_b) \right\|_2^2. \tag{3.14}$$

Of course, one needs to make a trade-off between these two objectives. The overall loss function is then:

$$\mathcal{L}_{\mathrm{DAAE}}(\mathcal{S}; \theta_z, \theta_b, \theta_e) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \ell_b^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)}, \boldsymbol{x}_{\mathrm{B}}^{(i)}; \theta_z, \theta_b) - \lambda \ell_e^{(i)}(\boldsymbol{x}_{\mathrm{A}}^{(i)}, \boldsymbol{x}_{\mathrm{E}}^{(i)}; \theta_z, \theta_e) \tag{3.15}$$

Specifically, the weights are updated in the following way:

$$\theta_z \leftarrow \theta_z - \mu \left(\frac{\partial \ell_b^{(i)}}{\partial \theta_z} - \lambda \frac{\partial \ell_e^{(i)}}{\partial \theta_z} \right), \tag{3.16}$$

$$\theta_b \leftarrow \theta_b - \mu \left(\frac{\partial \ell_b^{(i)}}{\partial \theta_b} \right), \tag{3.17}$$

$$\theta_e \leftarrow \theta_e - \mu \left(\frac{\partial \ell_e^{(i)}}{\partial \theta_e} \right), \tag{3.18}$$

where μ represents the learning rate. However, due to the presence of the parameter λ in (3.16), SGD is not applicable. This problem can be solved by the GRL: during forward propagation this layer is transparent, i.e., it just copies the input to its output; during backward propagation the GRL obtains the gradient from the subsequent layer and multiplies it by $-\lambda$ before it is passed to the previous layer. Formally, the GRL can be defined by the following set of equation:

$$R_\lambda(\boldsymbol{x}) = \boldsymbol{x}, \tag{3.19}$$

$$\frac{dR_\lambda}{d\boldsymbol{x}} = -\lambda \mathbf{I}, \tag{3.20}$$

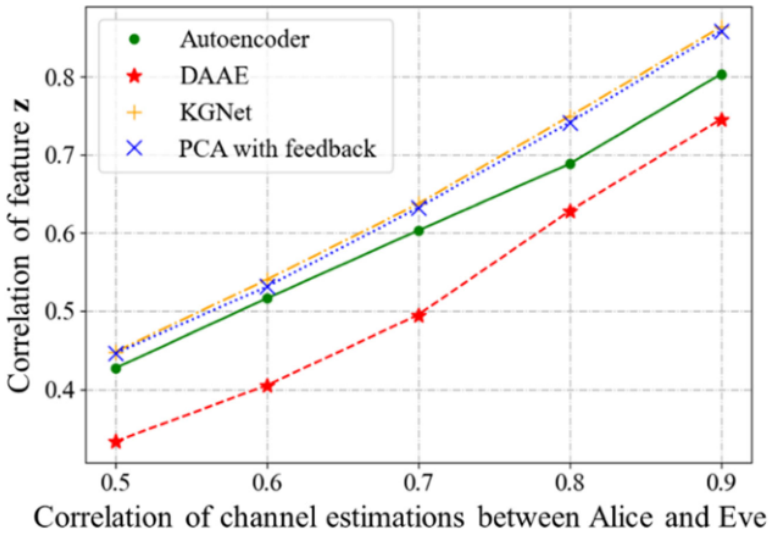


Figure 3.11: Correlation coefficient of the feature $\boldsymbol{z}$ between Alice and Eve [4].

where $\mathbf{I}$ is the identity matrix.

As it has been done for the MTAE, also here the authors evaluate the Pearson correlation coefficient ρ between the extracted features and inspect the latent space via t-SNE. This time, besides PCA and the AE baseline, they compare the DAAE also with the *Key Generation neural network* (KGnet) architecture [14], where it is assumed that Eve's distance is greater than half a wavelength and so KGnet does nothing to protect against it. From Fig. 3.11 is clear that the DAAE yields a lower correlation between Alice and Eve extracted features when Eve is really close to Alice with respect to the other three architecture. This differentiation can also be seen from Fig. 3.12, where $\boldsymbol{z}_{\mathrm{A}}$ (yellow dots) and $\boldsymbol{z}_{\mathrm{E}}$ (purple dots) are far apart in the two-dimensional space.

3.2.2 Machine Learning implementations

Apart from CIR, CFR, and RSS, it is also possible to design an algorithm on the basis of other kinds of features. For example, the authors in [15] propose a SKG procedure exploiting the *Inter-Pulse Interval* (IPI) of ECG signals. Here Alice and Bob are two sensors belonging to the same *Wireless Body Area Network* (WBAN), which collect ECG signals from two different locations. The authors developed a quantizer that has in input Alice and Bob IPI sequences x and y, and quantizes such values in a joint manner. In particular, let Q_x and Q_y be Alice and Bob's quantizers, and $\tilde{x} = Q_x(x)$ and $\tilde{y} = Q_y(y)$ the quantized versions of x and y. Of

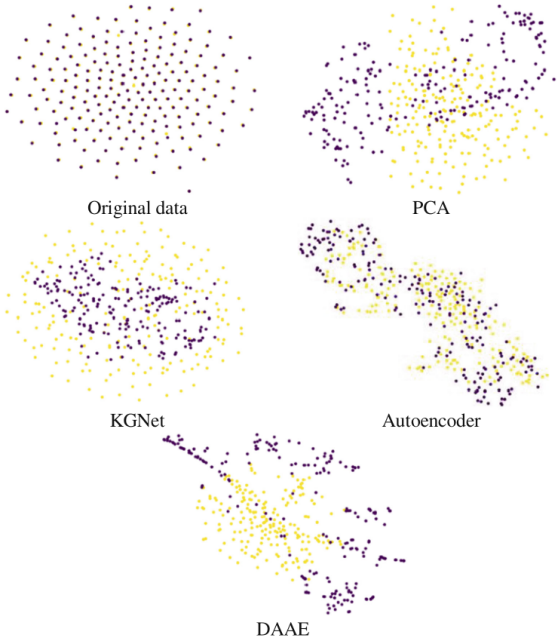


Figure 3.12: t-SNE visualization of features [4].

course, $\tilde{x}, \tilde{y} \in \{1, \ldots, \mathrm{R}\}$, with R the number of regions.

The best situation is when the two quantized sequences coincide, thus one can define a random variable x^* as:

$$x^* = \begin{cases} \tilde{x} & \text{if } \tilde{x} = \tilde{y} \\ \emptyset & \text{if } \tilde{x} \neq \tilde{y} \end{cases} \tag{3.21}$$

Then, the *Probability Mass Distribution* (PMD) of x^* is:

$$p_{x^*}(r) = \int_{R_r^x \times R_r^y} p_{xy}(a, b)\, da\, db, \quad r \in \{1, \ldots, \mathrm{R}\}, \tag{3.22}$$

where R_r^x and R_r^y are the Alice and Bob's quantizer regions associated to $r \in \{1, \ldots, \mathrm{R}\}$, while the probability of non-coincidence is $P[x^* = \emptyset] = 1 - \sum_{r=1}^{\mathrm{R}} p_{x^*}(r)$.

The aim is to maximize the entropy times frequency of x^*, since we are not interested in the case where $\tilde{x} \neq \tilde{y}$, i.e.,

$$P[x^* \neq \emptyset]\mathrm{H}(x^*) = \sum_{r=1}^{\mathrm{R}} p_{x^*}(r) \log_{\frac{1}{2}} \left(\frac{p_{x^*}(r)}{P[x \neq \emptyset]} \right), \tag{3.23}$$

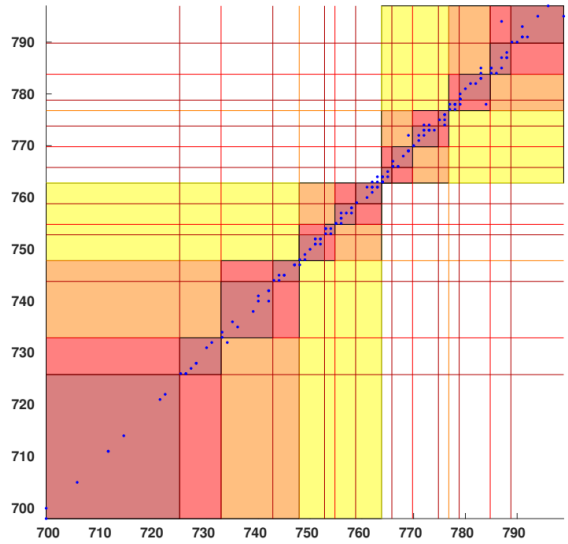


Figure 3.13: IPI quantization thresholds for 1- (yellow), 2- (orange), 3- (red), and 4-bit (dark red) quantization. Values are in ms.

where $p_{x^*}(r) = \frac{N_r}{N}$ is the empirical PMD of x^*, N_r is the cardinality of the r-th region, N is the total number of samples, and $P[x^* \neq \emptyset]$ is the probability that $\tilde{x} = \tilde{y}$, that is, Alice and Bob choose the same region. The equality in (3.23) comes from the fact that since the symbol $\emptyset$ is discarded, one has to scale the PMD by $P[x^* \neq \emptyset] = \sum_{r=1}^{\mathrm{R}} p_{x^*}(r)$.

The outcome of this non-uniform quantizer is shown in Fig. 3.13 for $\mathrm{b} \in \{1, 2, 3, 4\}$ quantization bits.

3.3 Considered channel features for SKG

From the CIRs gathered in a sea experiment carried out in Eilat, Israel in January 2022, we obtain the power-delay profile $H'_n(t, \tau)$ for the links Bob→Alice and Bob→Eve. Let $x_{i,n}(t)$ be the value of the i-th feature with $i = 1, \ldots, 4$, measured at time t by each node n. To extract the features, we zero out low-power arrivals in the power-delay profile, i.e.,

$$H_n(t, \tau) = \begin{cases} 0 & |H'_n(t, \tau)| < T_h \\ H'_n(t, \tau) & |H'_n(t, \tau)| \geq T_h. \end{cases} \tag{3.24}$$

Let $S_n(t)$ the set of delays of all channel arrivals that remain after this thresholding operation. Then, we consider the following four features for the SKA procedure [16]:

1. Number of channel taps

$$x_{1,n}(t) = |S_n(t)|\,, \tag{3.25}$$

2. Average tap power

$$x_{2,n}(t) = \frac{1}{|S_n(t)|} \sum_{\tau \in S_n(t)} |H_n(t,\tau)|\,, \tag{3.26}$$

3. Relative *Root Mean Square* (RMS) delay

$$x_{3,n}(t) = \left(\frac{1}{|S_n(t)| - 1} \sum_{\tau \in S_n(t), \tau \neq \tau_0} (\tau - \tau_0)^2 \right)^{1/2}, \tag{3.27}$$

4. Smoothed received power

$$x_{4,n}(t) = \alpha q_{n,t} + (1-\alpha)\, x_{4,n}(t'). \tag{3.28}$$

The experiment lasted 30 minutes, which is an insufficient time to collect enough data to train NNs. To overcome this issue, each data series $x_{n,k}$ has been fitted with a Gaussian *Kernel Density Estimation* (KDE), by estimating the PDFs $p_{x_{n,k}}(x)$. New correlated data can be generated with the following procedure [17]:

1. generate a $N \times K$ matrix of zero-mean correlated Gaussian variables $\tilde{v}$ with covariances

$$\mathrm{COV}(v_{n,k}, v_{n',k'}) = \begin{cases} 1 & \text{if } n = n' \text{and } k = k' \\ \alpha & \text{if } n \neq n' \text{and } k = k' \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

   where $\alpha \in [0,1]$ is the parameter controlling the covariance between nodes.

2. compute $u_{n,k} = F_x^N(\tilde{v}_{n,k})$, where $F_x^N(\tilde{v}_{n,k})$ is the CDF of a Normal distribution, and derive $x_{n,k} = F_{x_{n,k}}^{-1}(u_{n,k})$ via numerical methods.

4

Quantizer

The quantization phase is essential to map real values, i.e., the pre-processed channel measurements, into a stream of bits, representing the initial key. This quantization operation should ideally yield a sequence of independent and identically distributed bits, otherwise an eavesdropper could exploit the correlation between symbols to infer future values.

We want to design a joint quantizer that takes in input both Alice's and Bob's channel features $\boldsymbol{x}$ and $\boldsymbol{y}$ in such a way that each of them can independently generate the primary key by only considering their measurements. In particular, let us consider the case where Alice and Bob only compute one channel feature: these measurements can be arranged on the x axis and the y axis, respectively, thus obtaining a two-dimensional space. We aim at building an algorithm capable of dividing both axis into a given number of intervals, in such a way that, after this thresholding operation, Alice and Bob are capable of independently generate the initial key by only looking in which bin their measurements fall into. We tackle this problem by grouping the data points inside the joint Alice-Bob input space into rectangle-shaped clusters, hence obtaining a grid-like arrangement, and by ensuring that we select only one of this regions for each row and column, to avoid ambiguities when each user will look only at their intervals.

In formulas, we have built two quantizers Q_x and Q_y, respectively identified by the chosen thresholds on the abscissa and the ordinate, that Alice and Bob will use to extract the primary

key from their measurements. Specifically, we assign to each of the chosen regions a symbol as:

$$\begin{aligned} \tilde{x} &= Q_x(x), \quad \forall x \in \mathcal{X} \\ \tilde{y} &= Q_y(y), \quad \forall y \in \mathcal{Y}, \end{aligned} \tag{4.1}$$

where $\tilde{x}, \tilde{y} \in \{0, \ldots, \mathrm{R}-1\}$, with R the number of regions, are the quantized values for a particular measurement of Alice and Bob, respectively, $\mathcal{X}$ is the vector containing the realizations of Alice's channel feature, and $\mathcal{Y}$ is the vector containing the realizations of Bob's channel feature. Ideally, we would like to have $\tilde{x} = \tilde{y}$ and to do so, we need to train the algorithm accordingly. How to do the training is explained in the subsequent sections.

4.1 Correlated Gaussian random variables

As an example, we consider here the case where the observations of Alice and Bob are two correlated Gaussian random variables. If we collect a reasonable number of realizations, these will "be contained" in an ellipse. To simulate a pair of correlated normal random variables, start by generating independently the observation of Alice as:

$$x \sim \mathcal{N}(0, 1) \tag{4.2}$$

and another random variable $n \sim \mathcal{N}(0, 1)$. Then, given a certain correlation ρ_{AB}, the observation of Bob can be generated as:

$$y = \rho_{AB} x + n\sqrt{1 - \rho_{AB}^2} \tag{4.3}$$

Let us generate, e.g., 10^3 pairs (x, y) as in (4.2) and (4.3) and plot them in the two dimensional plane as in Fig. 4.1. The points are indeed aligned along an ellipse, the width of which is controlled by the amount of correlation, namely the parameter ρ_{AB}. The first version of the proposed quantizer is developed with the assumption that Alice and Bob compute only one channel feature each. The idea is to build a two-dimensional space by putting Alice measurements along the first dimension, and Bob measurements along the second dimension. In this space, the algorithm organizes the finite region in which such measurements reside in an arbitrary number of rectangle-shaped clusters. As a first step, we can generate a data set as in (4.2) and (4.3) with, e.g., $\rho_{AB} = 0.9$.

We would like to split such data set with rectangle-shaped regions that have, on average, the

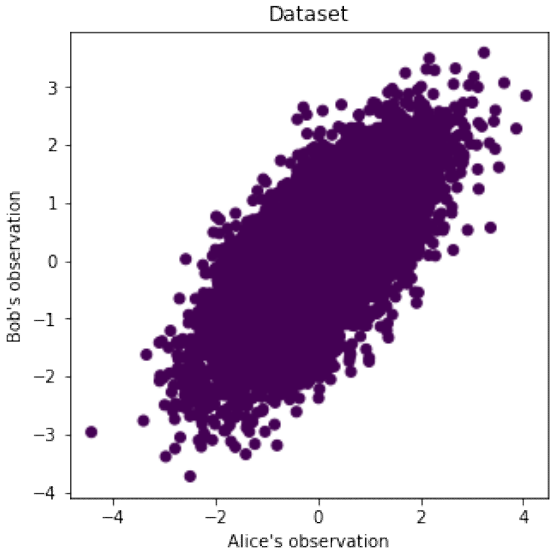


Figure 4.1: Plot of Alice and Bob realizations with $\rho_{AB} = 0.7$.

same number of points inside them but at the same time we also want not to discard too many samples. To achieve such result, the algorithm has been developed in the following way:

1. Drop a cross randomly within the region, defining four sub-regions;

2. If the number of points inside the two sub-regions on the diagonal is larger than the number of points inside the two sub-regions on the anti-diagonal, choose the first, otherwise the latter;

3. Search the best position for the cross (identified by current Alice and Bob's thresholds), that is, the position that minimizes the number of points left outside the two chosen sub-regions and maximizes the uniformity of the two regions;

Once the two best sub-regions are identified, the above reasoning can be repeated for both of them in a recursive manner, until we get the required number of rectangles. This process is detailed in Algorithm 4.1.

Algorithm 4.1 Developed joint non-uniform quantizer.

```
input data set S = (X, Y)
Require in input R even
n_crosses ← R − 1
n ← 0
label each point in S to -1 (unassigned)
step ← 0.1
ε ← 10^−6
while true
    split S on the basis of the label
    for each split
        cross_position ← random sample in current region
        best_cross_position ← cross_position
        best_loss ← inf
        best_loss_old ← best_loss
        loss_old ← inf
        while True
            try moving cross making a step upleft, upright, downleft, and downright
            re-compute regions for each movement
            save loss value for each movement
            choose move with minimum loss value
            if best move loss < loss_old
                move cross to that position
                best_loss ← best move loss
            else
                do not move cross
            if best_loss = best_loss_old
                step ← step − 10^−3
            if step < ε
                step ← 0.1
                break
            best_loss_old ← best_loss
        save best cross position and label samples according to it
        n ← n + 1
        if n = n_crosses
            return the labeled data set S
    update data set S with labels yielded by found cross positions
```

4.1.1 Local objective function

For this algorithm to work, one needs to define a proper metric to minimize or maximize. The following objective function has been implemented:

$$\mathcal{L}(S, r_1, r_2) = \alpha \sum_{s \in S} \mathbb{1}\{s \notin r_1 \wedge s \notin r_2\} + (1 - \alpha) \sum_{i=1}^{2} |N_i - \bar{N}| \tag{4.4}$$

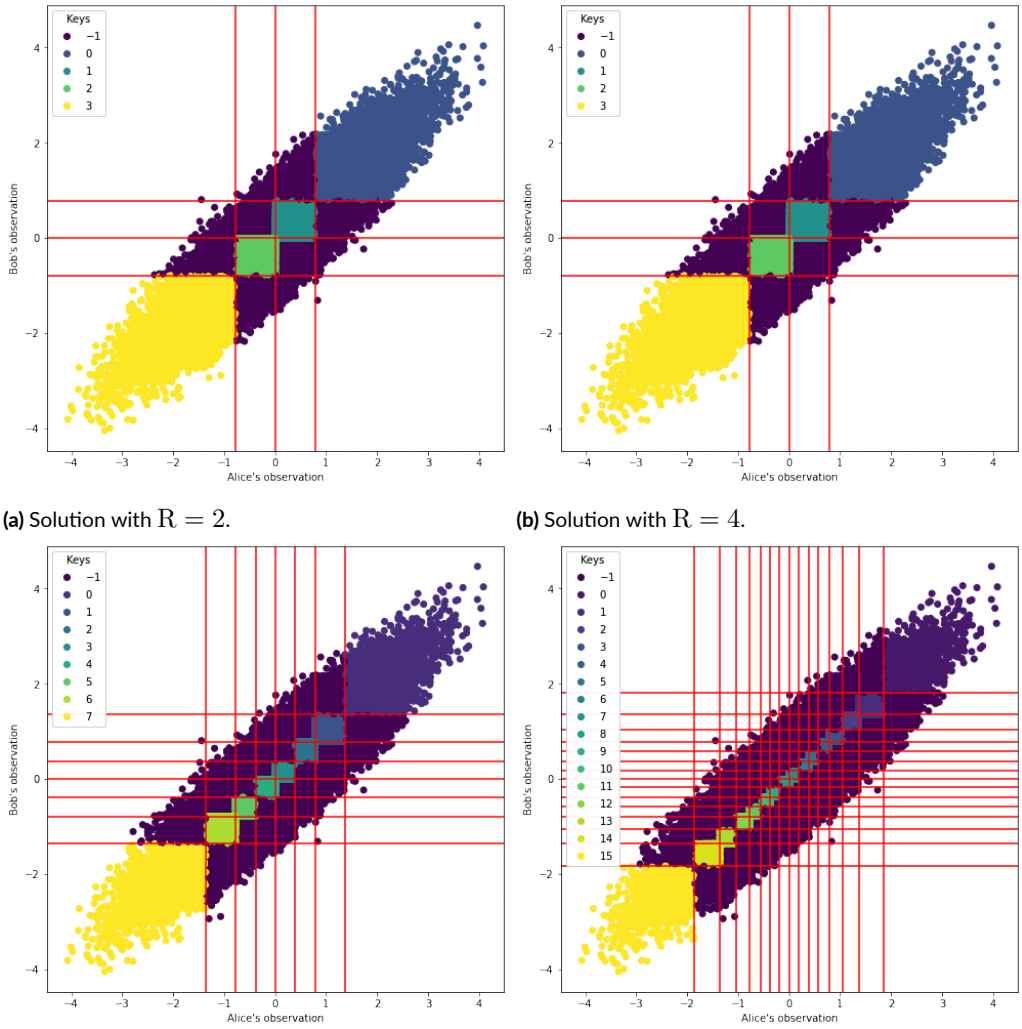
where $\mathbb{1}\{\cdot\}$ is the indicator function, N_i is the cardinality of the i-th region, and $\bar{N}$ is the average cardinality of the regions r_1 and r_2. It should be noted that this objective function operates locally, that is, it considers only the points within the current subset of the data set. The first term represents the number of points left outside with the current choice of regions, while the second term measures the non-uniformity of the two regions, that is, how much their cardinality is far from the average. These two requirements represent two plates of a scale, hence the parameter α allows to control the trade-off. In Fig.4.2 are shown the solutions for $\mathrm{R} = \{2, 4, 8, 16\}$ regions with $\alpha = 0.5$.

After the joint training on both Alice and Bob observations, the first coordinate of the dropped crosses divides the first axis in a certain number of intervals, and the same does the second coordinate of the placed crosses for the second axis. The result is a grid with different cell size, and one can notice that for each row and column, there exist only one region. Hence, each party obtains the initial key by only looking at their observation. In particular, they can independently quantize their measurements by looking in which interval these fall into. Ideally, both measurements of both parties should fall inside the same interval, i.e, they choose the same region, and so the same symbol. In this way, the information reconciliation phase can be avoided and Alice and Bob would already have a secret key that can be used right away.

To assess the performance of this algorithm, we can estimate the *Key Agreement Probability* (KAP) as in Algorithm 4.2, defined as:

$$\mathrm{KAP} = \mathbb{P}[\mathrm{K_A} = \mathrm{K_B}] \tag{4.5}$$

where $\mathrm{K_A}$ and $\mathrm{K_B}$ are the keys distilled by Alice and Bob, respectively. For comparison, a uniform quantizer, that simply divides each coordinate in intervals of the same length, has been built. In Table 4.1 are reported the outcomes with $\mathrm{R} = \{2, 4, 8, 16\}$ regions of the developed quantizer and the uniform quantizer, respectively. Here it seems that the latter is better, since it retains a higher KAP with an increasing number of regions.



(a) Solution with $\mathrm{R} = 2$.

(b) Solution with $\mathrm{R} = 4$.

(c) Solution with $\mathrm{R} = 8$.

(d) Solution with $\mathrm{R} = 16$.

Figure 4.2: Solutions with $\alpha = 0.5$

Algorithm 4.2 Key Agreement Probability Estimation

take in input Alice and Bob thresholds
$iters \leftarrow 10^6$
create $alice\_keys$ and bob_keys as empty arrays of length $iters$
for $i = 0; i < iters; i{+}{+}$
    generate a pair (x, y) of correlated Gaussian distributed random variables
    $j \leftarrow 0$
    while $x >$ j-th alice threshold
        $j \leftarrow j + 1$
    $alice\_keys[i] \leftarrow j$
    $j \leftarrow 0$
    while $y >$ j-th bob threshold
        $j \leftarrow j + 1$
    $bob\_keys[i] \leftarrow j$
output key agreement probability as $1 - \frac{\sum_{i=0}^{iters} alice\_keys[i] - bob\_keys[i]}{iters}$

Regions	**KAP**	
	Developed quantizer	Uniform quantizer
2	0.86	0.82
4	0.66	0.78
8	0.42	0.46
16	0.24	0.34

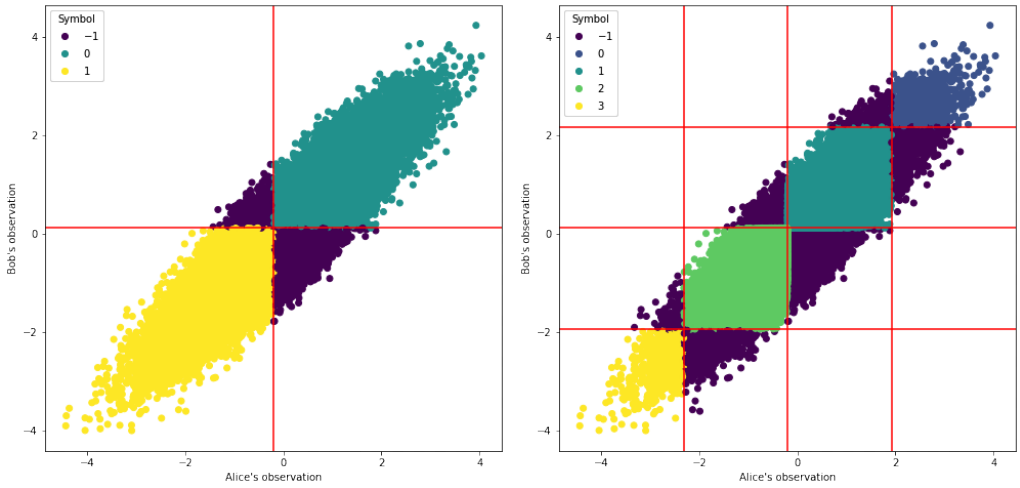
Table 4.1: KAP comparison between our solution and the uniform quantizer in function of the number of regions R.

However, by looking at how the uniform quantizer divides the data set (Fig. 4.3), the chosen regions are unbalanced for $\mathrm{R} > 2$, and in particular the inner regions contain much more points than the outer ones. This suggests that the symbols represented by the inner regions will be chosen with a higher probability than the others.

To confirm this assumption, we can compute the probability of each possible symbol, defined as:

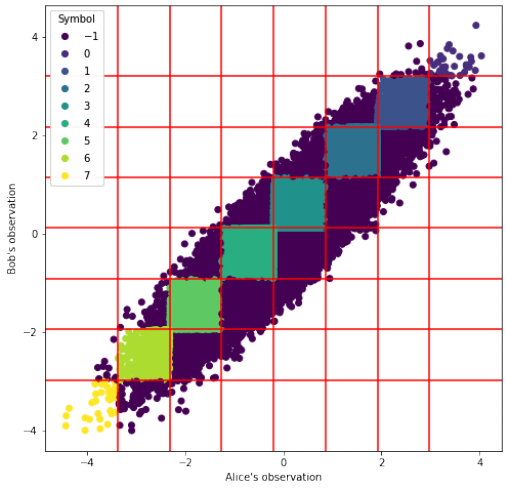
$$P_{k_i} = \mathbb{P}[\mathrm{K}_\mathrm{A}^{(i)} = k \wedge \mathrm{K}_\mathrm{B}^{(i)} = k | k = k_i] \tag{4.6}$$

by following Algorithm 4.3, for both our solution and the uniform quantizer. From the results shown in Fig. 4.4, one can indeed see that even though the uniform quantizer offers better KAP performance, the inner keys are selected with a much higher rate. An eavesdropper can thus neglect all the other keys and concentrate on the most probable ones. On the other hand, our solution yields a higher entropy, since the distribution of possible keys resembles a uniform

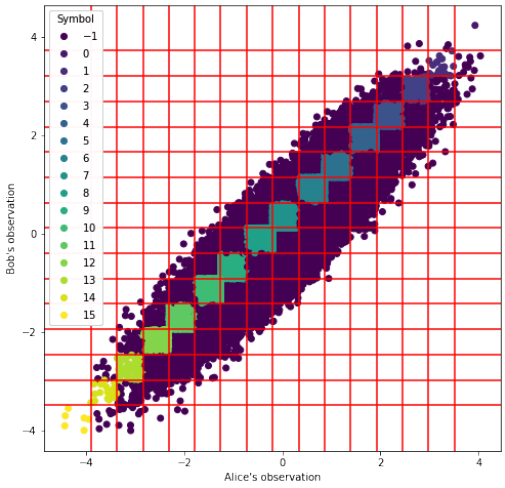


(a) Solution with $\mathrm{R} = 2$.

(b) Solution with $\mathrm{R} = 4$.



(c) Solution with $\mathrm{R} = 8$.



(d) Solution with $\mathrm{R} = 16$.

Figure 4.3: Joint uniform quantizer solutions.

distribution, i.e., each key is chosen with almost the same probability.

Algorithm 4.3 Estimation of probability for each possible key

input $alice\_keys$, bob_keys, and $possible\_keys$ arrays
$sum \leftarrow 0$
$N \leftarrow length(possible\_keys)$
create empty array $probs$ of length N
for $i = 0; i < N; i{+}{+}$
    $count \leftarrow 0$
    for $j = 0; i < length(alice\_keys); j{+}{+}$
        if $alice\_keys[j] = bob\_keys[j]$ and $alice\_keys[j] = possible\_keys[i]$ and $bob\_keys[j] = possible\_keys[i]$
            $count \leftarrow count + 1$
    $sum \leftarrow sum + \frac{count}{length(alice\_keys)}$
    $probs[i] \leftarrow \frac{count}{length(alice\_keys)}$
output $probs$

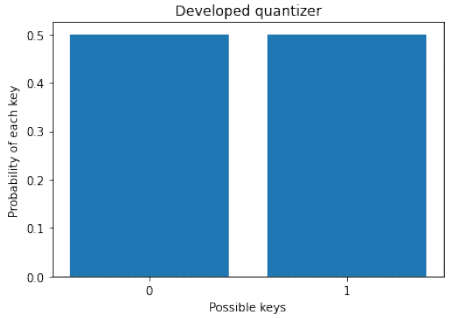
We want to investigate how the solution changes with different values of the trade-off parameter α. Let us define the normalized standard deviation as:

$$\sigma_{\text{norm}} = \frac{\sqrt{\frac{\sum_{i=1}^{\text{R}} (N_i - \bar{N})^2}{\text{R}}}}{\bar{N}} \tag{4.7}$$

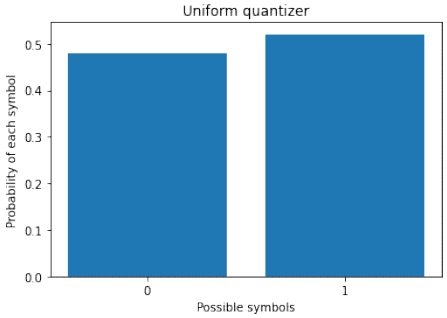
where R is the number of required regions and $\bar{N}$ is the average cardinality. Moreover, we define the exclusion probability as:

$$P_{\text{exclusion}} = \frac{\sum_{s \in \mathcal{S}} \mathbb{1}\{s \notin r_i \,,\, \forall i \in \{1, \ldots, R\}\}}{\sum_{s \in \mathcal{S}} \mathbb{1}\{s \notin r_i \,,\, \forall i \in \{1, \ldots, R\}\} + \sum_{s \in \mathcal{S}} \mathbb{1}\{s \in r_1 \vee \ldots \vee s \in r_R\}} \tag{4.8}$$

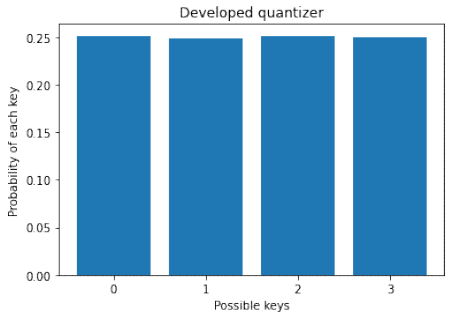
Then, by varying $\alpha \in \{0, 0.1, 0.2, \ldots, 1\}$ and by running the algorithm for each value, the graph in Fig. 4.5 is obtained. It compares the exclusion probability (orange curve) and the normalized standard deviation (blue curve) in function of α. If the exclusion probability curve is behaving more or less correctly, i.e., it becomes smaller if the algorithm is more and more penalized for points left outside the regions, the normalized standard deviation curve in a first moment grows, then decreases for middle values of α, and then goes up again. This is due to the fact that, despite having the parameter α controlling the trade-off between region uniformity and the count of excluded points, the objective function defined in (4.4) is local. If, e.g., one sets



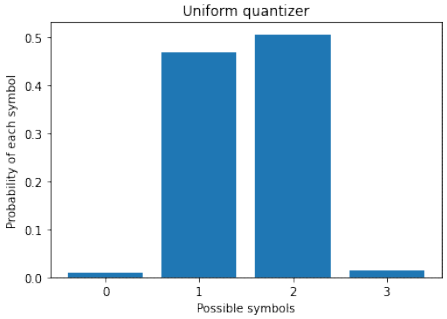
(a) Our solution with $\mathrm{R} = 2$.



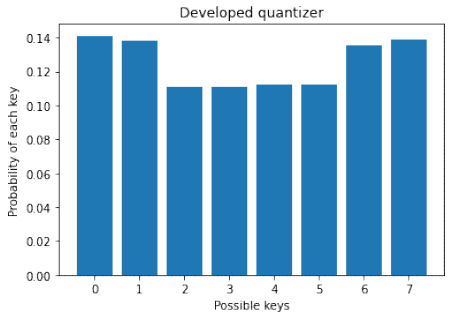
(b) Joint uniform quantizer with $\mathrm{R} = 2$.



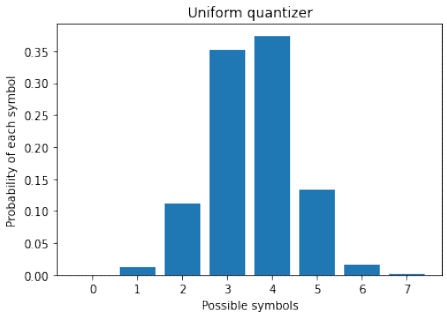
(c) Our solution with $\mathrm{R} = 4$.



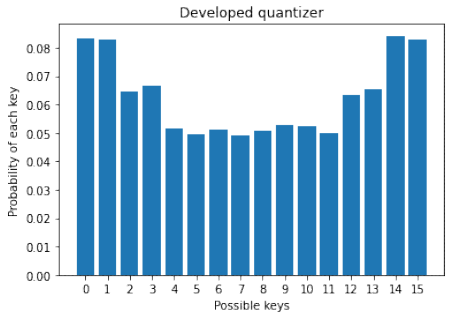
(d) Joint uniform quantizer with $\mathrm{R} = 4$.



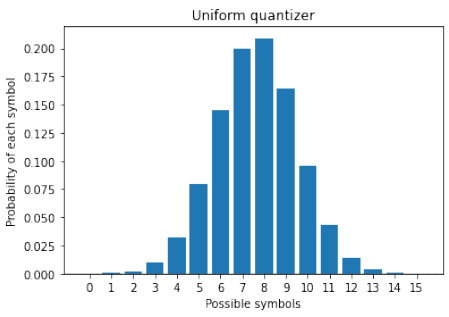
(e) Our solution with $\mathrm{R} = 8$.



(f) Joint uniform quantizer with $\mathrm{R} = 8$.



(g) Our solution with $\mathrm{R} = 16$.



(h) Joint uniform quantizer with $\mathrm{R} = 16$.

Figure 4.4: Comparison between key probabilities of our solution and the joint uniform quantizer.

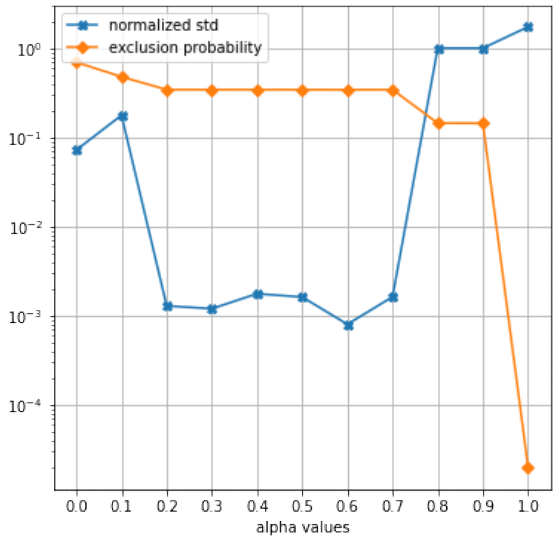


Figure 4.5: Normalized standard deviation versus exclusion probability in function of α with the local objective function and $\mathrm{R} = 4$.

$\alpha = 0$ the expected outcome is to have almost perfectly balanced regions, but the algorithm will minimize the objective function only in the portion of plane it is currently considering, thus reaching a local minimum in a certain subset of the data set which can be different from the local minimum obtained in another subset. The result is that the regions are uniform pair by pair. For this reason, we need a global function.

4.1.2 Eavesdropper presence

Let us consider the situation illustrated in Fig. 4.6, in which an eavesdropper, called Eve, is following Alice closely. The channel between Bob and Eve, namely h_{BE}, will be correlated with the legitimate channel h_{BA} and this goes in favor of Eve since she can observe values similar to what Alice observes. Moreover, we assume that the stalker knows the key extraction algorithm employed by legitimate parties, the thresholds that they use to independently quantize their measurements, and the channel features they are using for the SKG procedure. In particular, we assume that Eve manages somehow to obtain the thresholds computed by Alice and then, by computing the same feature that Alice and Bob are using, she extract her key by using Alice thresholds to quantize her measurements.

Unfortunately, as verified by the results in Section 6.1.1, if h_{BA} and h_{BE} are highly correlated Eve will find in most of the cases the same key of Alice, and thus Bob, since the key is symmet-

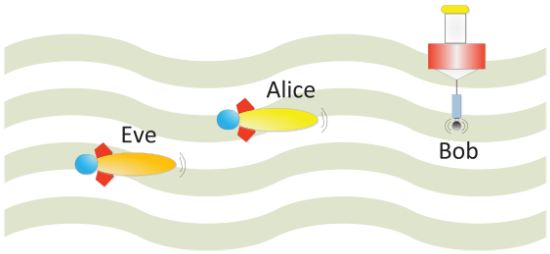


Figure 4.6: Stalking attack [5].

rical. In particular, the probability of Eve extracting the same key of Alice becomes larger and larger with a higher correlation factor ρ_{AE} between h_{BA} and h_{BE}, up to the point where Eve always (i.e., with probability 1) manages to obtain the same key as Alice, and thus Bob.

One can also see what is the actual *Secret Key Rate* (SKR) with Eve's presence. The SKR is defined as:

$$\mathrm{SKR} = \mathrm{I}(\tilde{x};\tilde{y}) - \mathrm{I}(\tilde{x};\tilde{z}) \tag{4.9}$$

Eve applies the same strategy of Alice and Bob with $\mathrm{R} = 2^{\mathrm{b}}$ regions – where $\mathrm{b} = \{2, 3, 4\}$ – and with ρ_{AE} varying in the interval $[0, 1]$, but she can only optimize her thresholds since the ones of Alice are given and constant.

For reference, we also plot the SKR for the joint uniform quantizer, where this time the thresholds on the x axis are found by uniformly splitting the projection on such axis of Alice and Bob data. From the results in Section 6.1.2 clearly see that our developed joint quantizer is capable of achieving way higher values in terms of SKR. This is because the uniform quantizer divides the input 2D space by splitting the first and second axis with intervals of the same length and thus is not capable of extracting all the possible information between Alice and Bob. Our joint quantizer instead produces non-uniform thresholds along the first and second axis in order to maximize the mutual information between Alice and Bob, resulting in a considerable upward shift in the SKR graph.

4.1.3 Adversarial training

The joint quantizer detailed in Algorithm 4.1 can be extended to include also the presence of the eavesdropper Eve. Consider the three-dimensional plane: we arrange Alice's measurements along the x axis, Bob's measurements along the y axis, and Eve's measurements along the z axis. The plane xy will still works as the original two-dimensional quantizer but now Alice and Bob have to optimize their thresholds also considering what Eve is doing. Specifically, the core

reasoning is the following:

1. the legitimate parties compute the value of the objective function in the four next possible locations of the cross and choose the next best move but do not perform the update yet;

2. Eve to make her decision on where to shift her threshold considering only her own measurements;

3. Alice and Bob then compute the objective function on the plane xz at the coordinates represented by the next best position of Alice's threshold and the new Eve's threshold chosen at point 2;

4. Alice and Bob take all the four saved values at point 1, update them by deducting the loss compute at point 3, and make the decision again on where to move the cross identified by their thresholds.

and it is repeated until convergence. Since Eve shift her threshold by only considering her measurements, she will only need to "see" how many points remains to the left and to the right of the threshold and move it (or choose not to, if it has already reached its optimal position) to the left or to the right by maximizing the uniformity of the two regions.

4.2 Generalization to higher-dimensional input

Suppose that Alice and Bob would like to use several features of all the available channel features for the SKA protocol. In this case, the input of the developed quantizer would lie in a higher dimensional space and in particular, we have:

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{\mathrm{L}\times 1} \tag{4.10}$$

where $\mathrm{L} \in \{2, 3, 4\}$ is number of features per user. This means that the quantizer input is 2L-dimensional and we cannot use the same strategy as in the two-dimensional case, in which for dividing each user input space a point is sufficient. Inspired by the *Support Vector Machine* (SVM) paradigm [18], we divide each user space with a hyperplane identified by the tuple $(\boldsymbol{a}_u, b_u)$, where $\boldsymbol{a}_u \in \mathbb{R}^{\mathrm{L}\times 1}$, $b_u \in \mathbb{R}$, and u indicates the user in consideration. To understand if a point $\boldsymbol{p}$ lies to the left or to the right of such hyperplane, a rule similar to the Hard-SVM

rule is applied, i.e.,

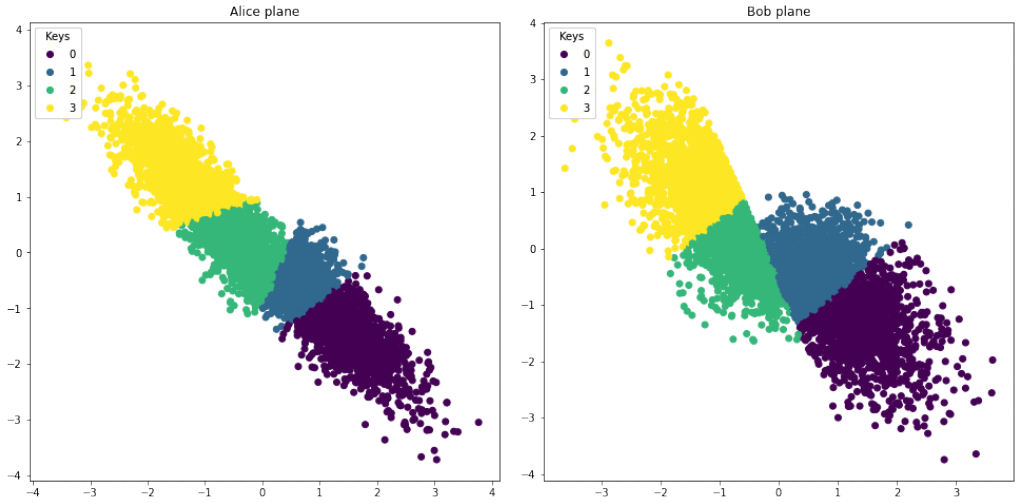
$$\text{the point } \boldsymbol{p} \text{ is } \begin{cases} \text{on the right} & \text{if } \boldsymbol{a}_u^T \boldsymbol{p} + b_u > 0 \\ \text{on the left} & \text{if } \boldsymbol{a}_u^T \boldsymbol{p} + b_u < 0 \end{cases} \tag{4.11}$$

Since we do not have a differentiable objective function based on the hyperplane values, we cannot apply gradient descent to update the hyperplane position. Thus, we opt for the following sub-optimal strategy to divide the plane of each party:

1. randomly sample a measurement of user u $\boldsymbol{m}_u \in \mathbb{R}^{\mathrm{L}\times 1}$ from the data set;
2. get one of the many hyperplanes passing through $\boldsymbol{m}_u$ by initializing $\boldsymbol{a}_u$ randomly and setting $b_u = -\sum_i m_u^i$;
3. optimize the hyperplane by shifting it to the left or to the right by updating the bias b_u and by changing the slope by updating the vector $\boldsymbol{a}_u$, on the basis of the objective function (6.1).

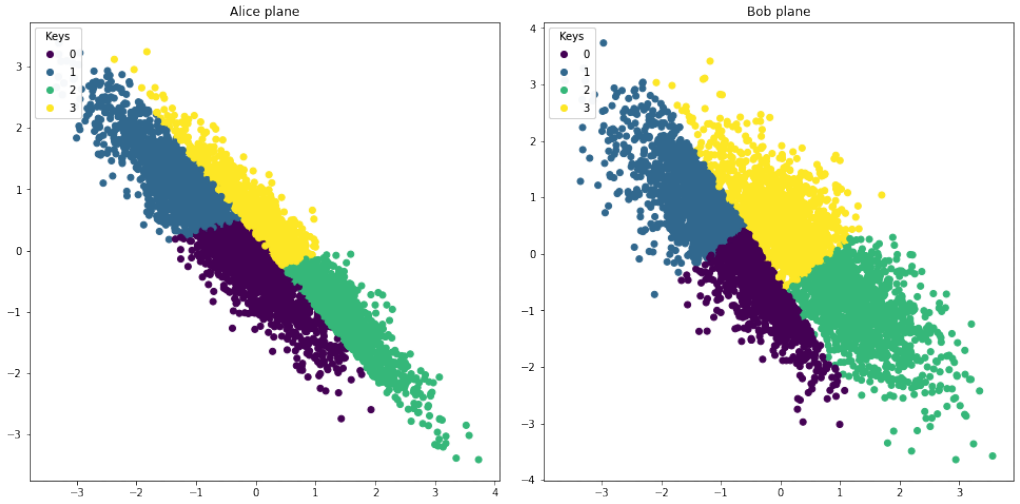
In other words, the hyperplanes of Alice and Bob form a higher dimensional cross and we can indeed reuse the Algorithm 4.1 with the modifications listed above. Also here, the optimization of the hyperplane is done multiple times to avoid bad local minima, that is, the strategy above is repeated a certain number of times and we choose the hyperplane yielding the best loss.

With $\mathrm{L} > 2$ the input space of the quantization algorithm we cannot visualize the results anymore. However, if $\mathrm{L} = 2$ the measurement of each user lies in a two-dimensional space and we can see how the algorithm divides such planes. In Fig. 4.7 and Fig. 4.8 are shown two possible solutions for $\mathrm{R} = 4$ and $\alpha = 0$ projected onto Alice and Bob's input spaces: both are divided into four parts by the projections of the optimized hyperplanes – which are lines – and if we intersect each part of Alice's plane with the appropriate one in Bob's plane we get the correspondent region in the 2L-dimensional space. The unassigned points were omitted from the plot as they were overlapping with the classified points because of the dimensionality reduction.



(a) Alice input space. **(b)** Bob input space.

Figure 4.7: One possible solution of the generalized algorithm with $\mathrm{L} = 2$ and $\mathrm{R} = 4$.



(a) Alice input space. **(b)** Bob input space.

Figure 4.8: Another possible solution of the generalized algorithm with $\mathrm{L} = 2$ and $\mathrm{R} = 4$.

5

Secret Key Generation With Autoencoders

5.1 The autoencoder

An autoencoder (AE) is a neural network (NN) composed by an encoder, whose task is to project the input space to lower-dimensional space, often called *latent space*, and a decoder, whose task is to recover the input from the compressed information. Since the latent space has a smaller dimension with respect to the input, i.e., it represents a bottleneck, the reconstruction will not be perfect. However, in this way the AE is forced to learn the most useful properties of input data.

Let n and m be the dimensions of input data $\mathcal{S}$ and latent space, respectively, with $n > m$. Let us define the encoder as a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and the decoder as a function $g : \mathbb{R}^m \to \mathbb{R}^n$. Then, the output of the AE is:

$$\hat{\boldsymbol{x}} = g(f(\boldsymbol{x}; \theta_{enc}); \theta_{dec}), \quad \boldsymbol{x} \in \mathcal{X} \tag{5.1}$$

where $\theta_{enc}, \theta_{dec}$ are the encoder and decoder weights, respectively.

Learning occurs by minimizing an appropriate loss function. Typically, for AEs this function is the MSE, also called the average reconstruction error:

$$\text{MSE} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \left\| \boldsymbol{x}^{(i)} - g(f(\boldsymbol{x}^{(i)}; \theta_{enc}); \theta_{dec}) \right\|_2^2 \tag{5.2}$$

5.2 System model

The setting is the following. As we know Alice and Bob are the two legitimate parties that want to generate a symmetric secret key, while Eve is the eavesdropper which tries to discover the key. We have a data set $\mathcal{S}$ containing the occurrences of four channel features: we denote with $\boldsymbol{x}_{\mathrm{A}}$ Alice data, with $\boldsymbol{x}_{\mathrm{B}}$ Bob data, and with $\boldsymbol{x}_{\mathrm{E}}$ Eve data.

The original data set $\mathcal{S}$ is split into a training (70%) set and a test (30%) set. Let the training set be $\mathcal{S}_{\mathrm{train}}$ and the test set be $\mathcal{S}_{\mathrm{test}}$.

We first make use of synthetic channel features, i.e., each channel feature of each user is generated according to a standard Gaussian distribution, and then we will repeat the experiments with the underwater measurements detailed in Section 3.3.

5.3 Autoencoder with uniform quantizer

We make use of an AE to reconstruct Bob's features, $\boldsymbol{x}_{\mathrm{B}}$, starting from Alice's features, $\boldsymbol{x}_{\mathrm{A}}$. Hence, we are dealing with a denoising AE. In particular, the encoder f is composed of a single fully-connected layer of size 4, the latent space is of size 2, and the decoder g is again a single fully-connected layer of size 4.

The training is done by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}_{\mathrm{AE}}(\mathcal{S}_{\mathrm{train}};\theta_f,\theta_g) &= \frac{1}{|\mathcal{S}_{\mathrm{train}}|}\sum_{i=1}^{|\mathcal{S}_{\mathrm{train}}|}\left\|\boldsymbol{x}_{\mathrm{B}}^{(i)}-\hat{\boldsymbol{x}}_{\mathrm{B}}^{(i)}\right\|^2 \\ &= \frac{1}{|\mathcal{S}_{\mathrm{train}}|}\sum_{i=1}^{|\mathcal{S}_{\mathrm{train}}|}\left\|\boldsymbol{x}_{\mathrm{B}}^{(i)}-g(f(\boldsymbol{x}_{\mathrm{A}}^{(i)};\theta_f);\theta_g)\right\|^2\end{aligned} \tag{5.3}$$

Specifically, we have trained the model with the `PyTorch` framework with the `Adam` optimizer for 250 epochs and a learning rate $lr = 2\cdot 10^{-4}$.

After training, the encoder f followed by a uniform quantizer (UQ) Q is used to produce the initial keys of Alice and Bob, and in particular:

$$\begin{aligned}\boldsymbol{z}_{\mathrm{A}} &= Q(f(\boldsymbol{x}_{\mathrm{A}})) \\ \boldsymbol{z}_{\mathrm{B}} &= Q(f(\boldsymbol{x}_{\mathrm{B}}))\end{aligned} \tag{5.4}$$

where $\boldsymbol{z}$ is the latent vector produced by the inputs $\boldsymbol{x}$ of each user.

This architecture has two main issues:

1. the symbols composing the initial keys are not equally likely, and this translates into poor key randomness that can be exploited by an eavesdropper;
2. we do not take into account the presence of an eavesdropper, and in fact the SKR

$$\mathrm{SKR} = \mathrm{I}(\boldsymbol{z}_{\mathrm{A}}; \boldsymbol{z}_{\mathrm{B}}) - \mathrm{I}(\boldsymbol{z}_{\mathrm{A}}; \boldsymbol{z}_{\mathrm{E}}) \tag{5.5}$$

goes to zero when $\rho_{\mathrm{AE}} = \rho_{\mathrm{AB}}$.

5.4 Autoencoder with Differentiable Soft Quantization

We now investigate the effects of training the AE in Section 5.3 with a quantization layer directly embedded between encoder and decoder. However, this operation needs some adjustments since the learning through back-propagation cannot occur as the quantization function is not differentiable.

The proposed NN architecture is a fully-connected AE equipped with a *Differentiable Soft Quantization* (DSQ) layer [19] right after the latent space, but apart from this the architecture remains the same of the one described in Section 5.3.

The DSQ layer consists in a differentiable piece-wise function approximating the uniform quantizer. The DSQ function is implemented differently with respect to [19], and in particular it is defined as a finite sum of scaled and shifted `tanh` functions, i.e.,

$$Q(x) = \sum_{i=1}^{\mathrm{L}-1} \frac{\Delta}{2} \tanh\left(\kappa\left(x + \left(\frac{\mathrm{L}}{2} - 1\right)\Delta\right)\right) - \left(\frac{\mathrm{L}}{2} - 1\right)\Delta \tag{5.6}$$

where x (normalized between -1 and $+1$) is the signal to be quantized, $\mathrm{L} = 2^{\mathrm{b}}$ is the number of levels required, with b the quantization bits, $\Delta = \frac{2}{\mathrm{L}}$ is the quantization step, while the coefficient κ is responsible for the steepness of transitions between each level. For example, in Fig. 5.1 we have the DSQ for $\mathrm{L} = 4$ (which means the step size is $\Delta = \frac{1}{2}$), and $\kappa = 100$. Given that the DSQ function is differentiable everywhere, it supports gradient back-propagation, and the network can now be trained directly with the quantization layer embedded between encoder

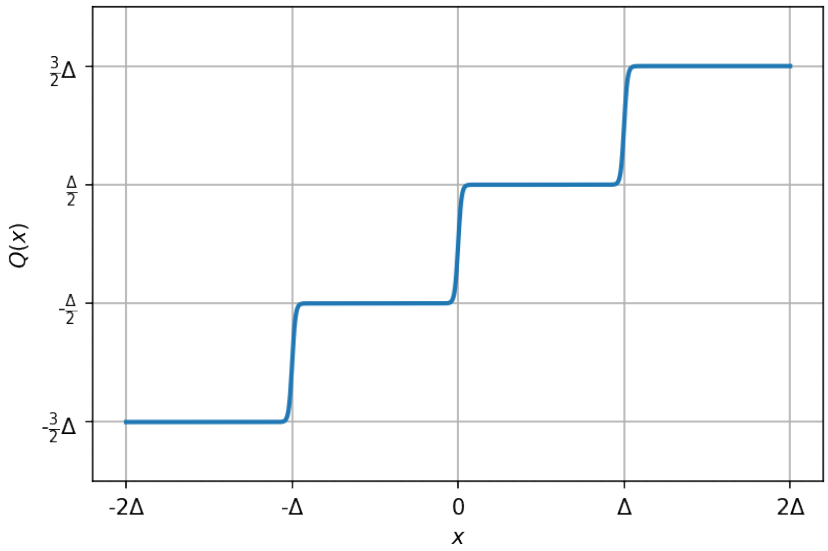


Figure 5.1: DSQ function with $\mathrm{L} = 4$ and $\kappa = 100$.

and decoder. Specifically, the model is trained with the same framework and parameters of the AE described in Section 5.3.

5.5 Eavesdropper presence

We have seen the effect of Eve's presence on the SKR when Eve's channel become more and more correlated with either Alice's or Bob's channel. Here we implement the MTAE and DAAE architectures (with appropriate and necessary modifications) described in Section 3.2 to train the AE in an adversarial-aware manner and thus to yield an initial key that contain few or (ideally) no information that Eve can exploit.

5.6 Enhance secret key uniformity

If the MTAE and DAAE architectures solve the problem of having the eavesdropper Eve at a distance less than half a wavelength, we still have been left with the problem that the symbols of the produced initial key are not equally likely. As it has been already pointed out, a poor key randomness facilitates the eavesdropper in discovering the key. To avoid this, we propose to use an architecture called *Adversarial Autoencoder* (AAE), which is employed to reshape the latent space distribution of an AE. As in Fig. 5.2, the AAE architecture consists in an encoder

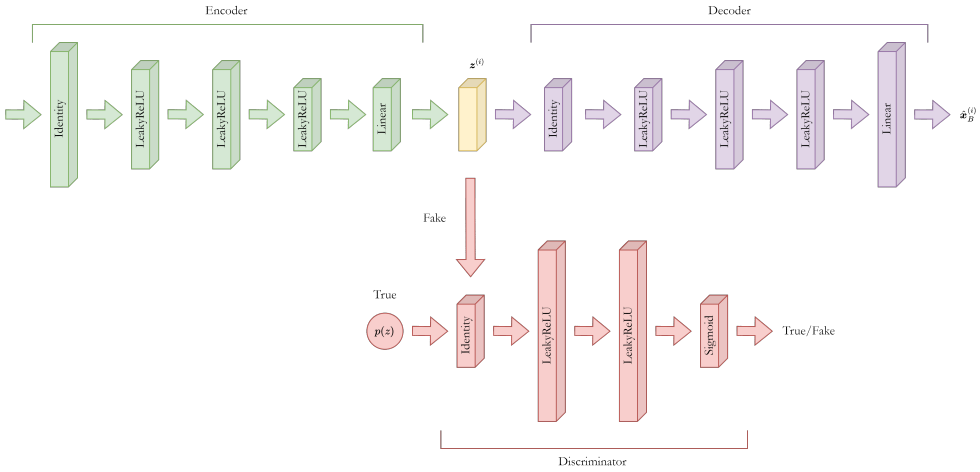


Figure 5.2: AAE architecture.

$f(\cdot;\theta_{enc})$, a decoder $g(\cdot;\theta_{dec})$, and a discriminator $h(\cdot;\theta_d)$. The encoder consists in five fully connected layer of sizes $4-3-3-2-2$, respectively. The inputs are just clamped to the first layer which then has an identity activation function, the following three layers have a LeakyReLU as activation function, whereas the last one has a linear activation. The decoder is the converse of the encoder and specifically it has four fully connected layers of sizes $2-2-3-3-4$, respectively, where the first one has an identity activation as it serves only to clamp take data in input, the next three lares are activated with a LeakyReLU, and the last one has a linear activation. Instead, the discriminator is made of four fully-connected layers of sizes $2-500-500-1$, respectively. The first layer has an identity activation function, the two hidden layers are activated with a LeakyReLU function, while the output layer has a sigmoid activation function.

Besides the classical AE task (i.e., reduction of the average reconstruction error between output and ground truth), we aim at forcing the encoder to output values following a given prior distribution $p(z)$. This is done by the discriminator, that tells if the data it sees comes from the "true" distribution or from the "fake" distribution, i.e., the encoder outputs. Clearly, the encoder must now be trained with two goals: to minimize the reconstruction error and to produce latent vectors that are distributed as close as possible to $p(z)$, in order to fool the discriminator.

The training of an AAE is hence done in two phases: (i) *reconstruction* phase and (ii) *regularization* phase.
The first is just what an AE would do and it is the same of what it is done in Section 5.3. For the second, we first train the discriminator to recognize if the data is coming from the prior distri-

bution or from the encoder, i.e., it should output 1 (True) if we pass random inputs according to $p(z)$ and 0 (Fake) if we pass the encoder outputs. This is accomplished by computing the [mean] binary cross-entropy loss (5.7) first by passing to the discriminator $\boldsymbol{z}_{\text{true}}$, sampled from $p(z)$), with targets fixed to $t^{(i)} = 1$, and then by passing $\boldsymbol{z}_{\text{fake}} = f(\boldsymbol{x}_{\text{A}}; \theta_{enc})$ with targets fixed to $t^{(i)} = 0$. For the second pass the weights θ_{enc} of the encoder are fixed as we do not want them to be updated in this phase.

$$\begin{aligned}\mathcal{L}_{\text{BCE}}(\boldsymbol{o}, \boldsymbol{t}) &= -\frac{1}{m}\sum_{i=1}^{m} t^{(i)} \log o^{(i)} + (1 - t^{(i)}) \log(1 - o^{(i)}) \\ &= -\frac{1}{m}\sum_{i=1}^{m} t^{(i)} \log h(\boldsymbol{z}^{(i)}; \theta_d) + (1 - t^{(i)}) \log(1 - h(\boldsymbol{z}^{(i)}; \theta_d))\end{aligned} \tag{5.7}$$

in which $o^{(i)}$ is the output of the discriminator, $t^{(i)}$ is the ground truth for a given i-th sample and m is the number of samples, and back-propagating the gradient only through the discriminator.

Then, we have to train the encoder to learn the required latent distribution: fix the discriminator weights θ_d (we have already trained it and do not want it to be updated in this phase) and the target to $t = 1$ (for the encoder the values that it produces are the real ones); forward encoder's generated data to the discriminator; compute the loss (5.7) and back-propagate the gradient only through the encoder.

The encoder and the discriminator are in opposition, i.e., the first tries to minimize the probability of its generated values to be flagged as Fake from the second , whereas the discriminator tries to maximize the probability of assigning the correct label to both the real data and the fake data.

The two goals of the AE are mutually-exclusive, that means, one must make a trade-off between how good the AE is at approximating Bob's features $\boldsymbol{x}_B$ and how good the encoder is at approximating the required latent distribution. This leads us to the definition of the overall loss function on which to train the AE:

$$\begin{aligned}\mathcal{L}_{\text{AAE}}(\mathcal{S}_{\text{train}}; \theta_{enc}, \theta_{dec}, \theta_d) = \lambda \mathcal{L}_{\text{AE}}(\boldsymbol{x}_{\text{A}}, \boldsymbol{x}_{\text{B}}; \theta_{enc}, \theta_{dec}) \\ + (1 - \lambda)\mathcal{L}_{\text{BCE}}(h(f(\boldsymbol{x}_{\text{A}}; \theta_{enc}); \theta_d), \mathbf{1})\end{aligned} \tag{5.8}$$

where the parameter λ controls the trade-off between the first and the second term. The latter is updated also by the discriminator, as explained earlier. The trade-off coefficient is empirically set to $\lambda = 0.2$ for the training with the correlated Gaussian data set, and to $\lambda = 0.1$ for the

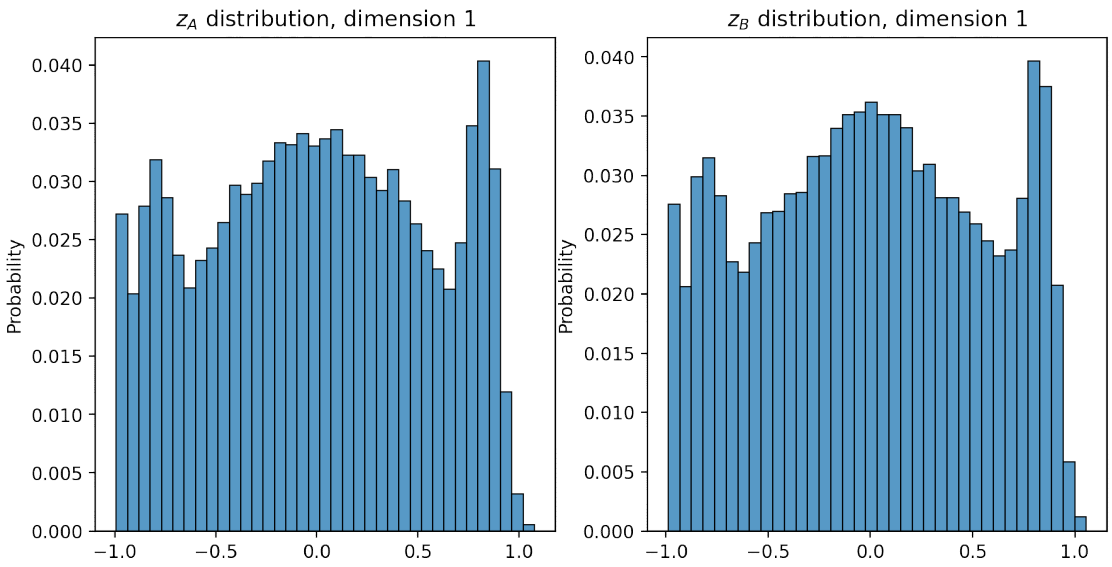


Figure 5.3: Latent space distributions for Alice's (left plot) and Bob's (right plot) primary keys, Gaussian data set.

training with the underwater data set.

Now we need to define the latent space distribution that we want the encoder to reproduce. To compare the performance of such architecture with the results in Section 5.3, a straightforward option is to have:

$$p(z) \sim \mathcal{U}(-1, 1) \tag{5.9}$$

If the AAE is trained with such $p(z)$, and the encoder is deployed on Alice and Bob as usual, we obtain the latent spaces in Fig. 5.3, both of them resembling a uniform distribution, as required. Thus, in the *exploitation phase* Alice and Bob can simply apply a uniform quantizer on top of their encoder to get their primary key.

5.6.1 Eavesdropper-aware training

To defend from an eavesdropper, we must train the AAE architecture in an eavesdropper-aware fashion, that is, we adopt the MTAE training method. More particularly, we add the second

term of (3.9) to the AAE loss function as a penalty term, obtaining:

$$\begin{aligned}\mathcal{L}_{\text{AAE}}(\mathcal{S}_{\text{train}};\theta_{enc},\theta_{dec},\theta_d) = \lambda\mathcal{L}_{\text{AE}}(\boldsymbol{x}_{\text{A}},\boldsymbol{x}_{\text{B}};\theta_{enc},\theta_{dec}) \\ + (1-\lambda)\mathcal{L}_{\text{BCE}}(h(f(\boldsymbol{x}_{\text{A}};\theta_{enc});\theta_d),\mathbf{1}) \\ - \beta\frac{1}{|\mathcal{S}_{\text{train}}|}\sum_{i=1}^{|\mathcal{S}_{\text{train}}|}\ell_z^{(i)}(\boldsymbol{x}_{\text{A}}^{(i)},\boldsymbol{x}_{\text{E}}^{(i)};\theta_z,\theta_b)\end{aligned} \tag{5.10}$$

with β, controlling the amount of the penalization, dynamically adjusted during training as in (3.8).

6 Numerical results

6.1 Gaussian data set

6.1.1 Developed joint quantizer

Hereinafter we come up with several global objective functions for the two-dimensional joint non-uniform quantizer described in Chapter 4 and present the results for each of them.

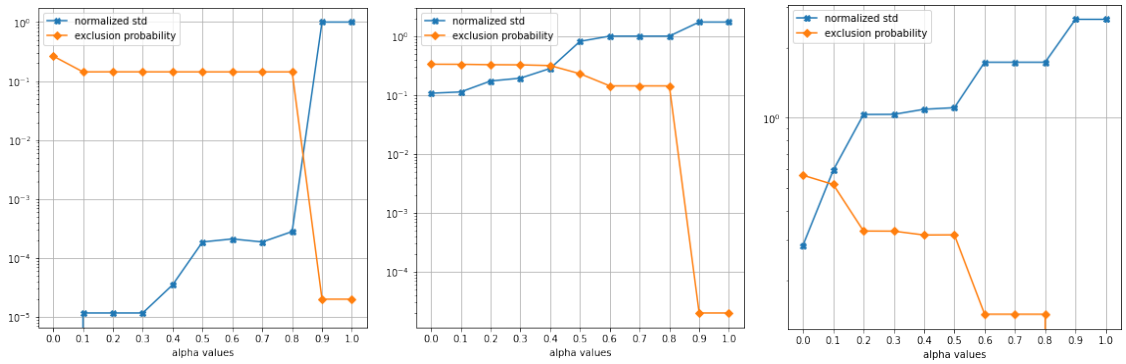
Global objective function with exclusion probability and normalized standard deviation

Although retaining the recursive nature of the algorithm, we let it have a sort of "memory", that is, in each iteration we save the optimal cardinalities found so that in the next iteration the algorithm can benefit from this information and it should reach a local minimum that minimizes the uniformity globally. Furthermore, we adapt the metrics used in the global function to the ones in (4.7) and (4.8), obtaining:

$$\mathcal{L}(\mathcal{S}, r_1, \ldots, r_{\mathrm{R}}) = \alpha P_{\text{exclusion}} + (1-\alpha)\sigma_{\text{norm}} \tag{6.1}$$

By running again the algorithm with different values of α as it has been done for the local function, Fig. 6.1 is obtained. Now the behavior of both curves is the desired one, and in particular the normalized standard deviation is increasing with α, whereas the exclusion probability de-

creases. Furthermore, both metrics start from a higher value with increasing number of regions and this is also expected because with a lower number of rectangles we can agglomerate a higher number of points and it is easier for the algorithm to balance less regions.

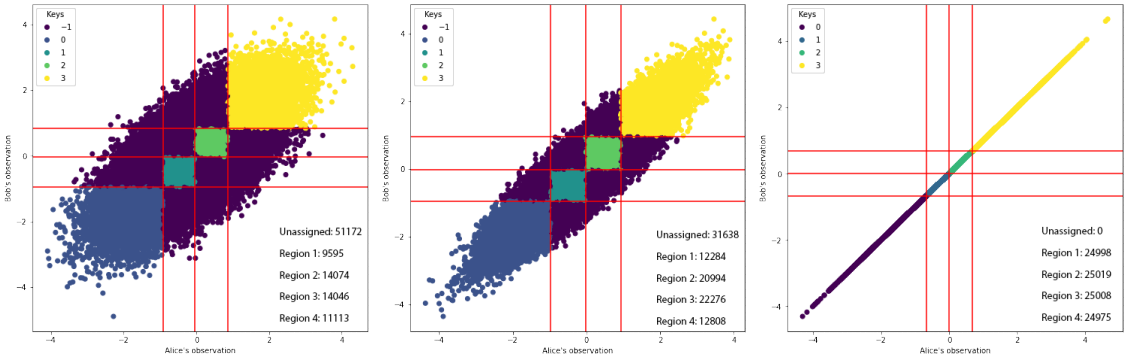


(a) With $\mathrm{R} = 2$. **(b)** With $\mathrm{R} = 4$. **(c)** With $\mathrm{R} = 8$.

Figure 6.1: Normalized standard deviation versus exclusion probability in function of α with the global objective function.

Global objective function with entropy

We have understood that, in the end, the goal is to maximize the entropy of the initial key, so we could rethink the objective function to optimize directly the entropy. Thus, we make use of the objective function (3.23) of [15] and run again our algorithm.



(a) $\rho_{AB} = 0.7$ **(b)** $\rho_{AB} = 0.9$ **(c)** $\rho_{AB} = 1$

Figure 6.2: Solutions with the entropy objective function, $\mathrm{R} = 4$ and different ρ_{AB} values.

From Fig. 6.2 one can notice that the algorithm equipped with the objective function in (3.23) works best when the data is highly correlated, that is, $\rho_{AB} \to 1$. In fact, if we consider

the outcome for four regions, the entropy reaches its maximum – namely $\mathrm{H}(x^*) = 2$ bits – with $\rho_{AB} = 1$, wheres already with $\rho_{AB} = 0.9$ the algorithm is not capable to reach such value. This means that the regions won't be so balanced. Instead, the algorithm equipped with our objective function, defined in (6.1), is not hindered by data correlation and it manages to create quite balanced regions with any value of ρ_{AB}, if we select $\alpha = 0$ for a correct comparison, but it leaves more points outside the regions.

Global objective function with mutual information

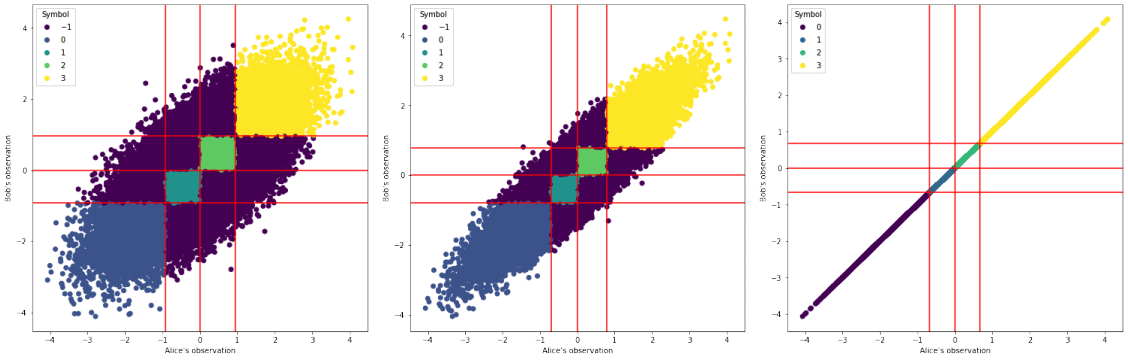
The objective function in (3.23) does not consider the non-classified points. Even though we would like to have balanced regions, the points left outside matter as they undermine the key agreement probability. In particular, given the thresholds of Alice and Bob at a certain optimization iteration, instead of considering the created regions as bins and building the empirical PMD by counting the points falling in each of them and dividing by the total number of points, we build the empirical joint PMD of Alice and Bob measurements by considering each grid cell as a bin. Moreover, we can also obtain the marginal PMDs of Alice and Bob by considering first only Alice's thresholds to create the bins, and then Bob's thresholds. We define the new objective function as the mutual information between Alice and Bob quantized measurements:

$$\mathrm{I}(\tilde{x};\tilde{y}) = \mathrm{H}(\tilde{x}) + \mathrm{H}(\tilde{y}) - \mathrm{H}(\tilde{x},\tilde{y}) \tag{6.2}$$

We want to understand how the algorithm behaves with different number of regions and correlation coefficient of channel features. Fig. 6.3 shows the solutions for different values of ρ_{AB} and it can be seen that the objective function based on the mutual information has the same problem of the objective function based on the entropy, that is, with highly correlated data the algorithm manages to achieve the maximum value, creating quite balanced regions, but with a larger σ^2_{AB} the resulting regions are not so balanced.

We analyze the convergence of such algorithm to check if it is behaving correctly. For the sake of clarity, let us assume $\rho_{\mathrm{AB}} = 1$. In this way it will be easier to recognize if the algorithm converges to the maximum of mutual information between Alice and Bob. As we can see from Fig. 6.4, for each thrown cross the algorithm manages to obtain the highest mutual information possible.

From Fig. 6.5 we see that with increasing correlation between the features of Alice and Bob the achieved mutual information tends to its maximum, namely $\mathrm{I}(\tilde{x};\tilde{y}) = \log_2 \mathrm{R}$. The experiment is repeated for $\mathrm{b} = 2, 3$, and 4 quantization bits, which means with $\mathrm{R} = 2^{\mathrm{b}} = 4, 8$, and



(a) $\rho_{\mathrm{AB}} = 0.7$ (b) $\rho_{\mathrm{AB}} = 0.9$ (c) $\rho_{\mathrm{AB}} = 1$

Figure 6.3: Solutions with the mutual information objective function, $\mathrm{R} = 4$ and different ρ_{AB} values.

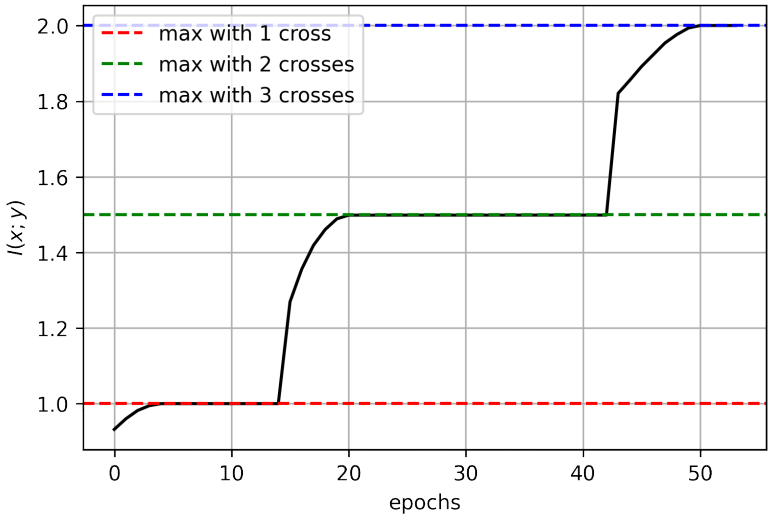


Figure 6.4: Convergence plot of the objective function (6.2).

16 regions in which the data is split.

Fig. 6.6 shows the KAP in function of ρ_{AE} of the developed joint quantizer. It can be seen that with, e.g., $\mathrm{b} = 2$, for $\rho_{\mathrm{AE}} = 0$ the KAP is just 0.25, since if Eve's channel is completely decorrelated from Alice's channel, trying to extract the key translates into the event $\mathcal{E} = \{$"Pick a number from 0 to R-1 randomly"$\}$, with R the number of regions which in this case is four. Therefore, the probability of such event will be $P(\mathcal{E}) = \frac{1}{\mathrm{R}} = \frac{1}{4}$.

The SKR, with $\mathrm{I}(\tilde{x}; \tilde{y})$ computed with $\rho_{\mathrm{AB}} = 1$, is shown in Fig. 6.7. It is clear that when the eavesdropper channel h_{BE} is perfectly correlated with the legitimate channel h_{BA}, the number of secret bits that Alice and Bob manage to extract is practically zero. Instead, if $\rho_{\mathrm{AE}} = 0$,

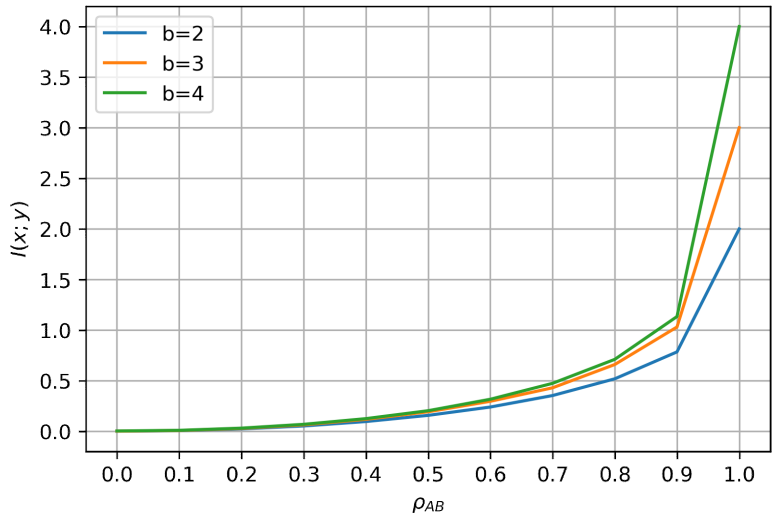


Figure 6.5: Mutual information between Alice and Bob for $\mathrm{b} = 2, 3, 4$ with the developed quantizer.

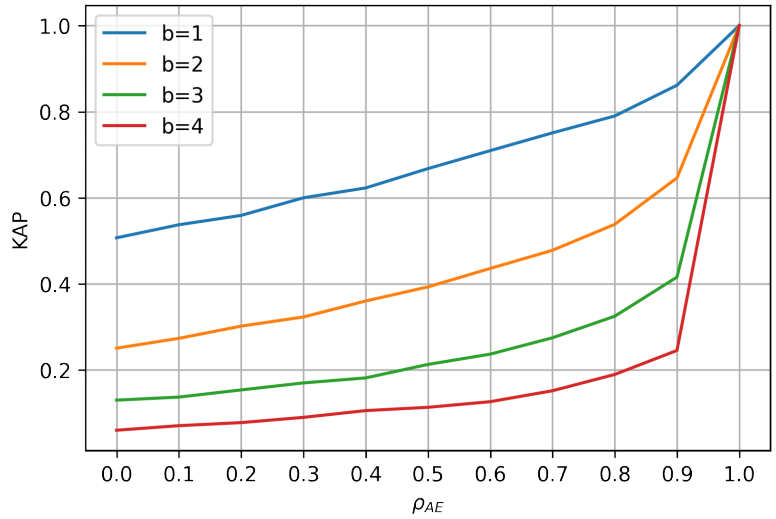


Figure 6.6: KAP between Alice and Eve for the developed quantizer, with $\mathrm{b} = 1, 2, 3, 4$ and $\rho_{\mathrm{AB}} = 0.9$.

i.e., Eve's measurements are completely decorrelated with respect to Alice, and since these are distributed according to a Gaussian then, Eve and Alice measurements are independent. Thus, the SKR becomes:

$$\mathrm{I}(\tilde{x}; \tilde{y}) - \mathrm{I}(\tilde{x}; \tilde{z}) = \mathrm{I}(\tilde{x}; \tilde{y}) \tag{6.3}$$

To obtain a plot spanning all SKR values we have assumed perfect correlation between Alice

and Bob, and (6.3) further simplifies into:

$$\mathrm{I}(\tilde{x};\tilde{y}) = \mathrm{H}(\tilde{x}) - \mathrm{H}(\tilde{y}|\tilde{x}) = \mathrm{H}(\tilde{y}) - \mathrm{H}(\tilde{x}|\tilde{y}) = \mathrm{H}(\tilde{x}) = \mathrm{H}(\tilde{y}), \tag{6.4}$$

due to the fact that two perfectly correlated Gaussian random variables are indeed exactly the same. Moreover, by applying the definition of entropy for discrete random variables, we can compute either $\mathrm{H}(\tilde{x})$ or $\mathrm{H}(\tilde{y})$ (since they are the same) in closed form. Specifically, we first recall the definition of entropy

$$\mathrm{H}(\tilde{x}) = -\sum_i p_i \log_2 p_i, \tag{6.5}$$

where $p_i = \mathrm{Q}(\frac{\mathrm{LB}_x^{(i)}}{\sigma_x}) - \mathrm{Q}(\frac{\mathrm{UB}_x^{(i)}}{\sigma_x})$ with $\mathrm{Q}(\cdot)$ defined as

$$\mathrm{Q}(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt \tag{6.6}$$

and $\mathrm{LB}_x^{(i)}$, $\mathrm{UB}_x^{(i)}$ being the i-th and $i+1$-th thresholds the projection of Alice and Bob data on the x axis has been divided into. These are found by running our joint quantizer algorithm with $\rho_{\mathrm{AB}} = 1$. The closed-form result for each quantization bit b is marked with a cross in Fig. 6.7, and it can be seen that it coincides perfectly with the maximum of each curve.

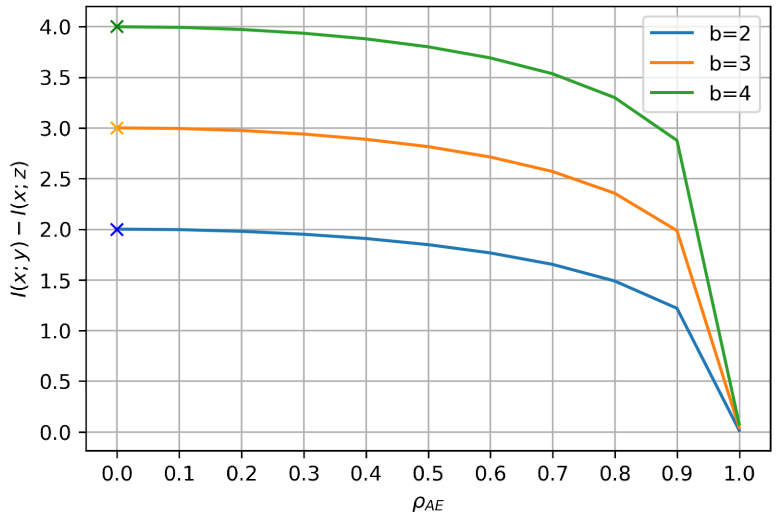


Figure 6.7: SKR in presence of an eavesdropper with our quantizer.

Adversarial training

We run the algorithm with a correlation between Alice and Bob measurements of $\rho_{\mathrm{AB}} = 0.9$ and with Eve's being correlated with Alice with a coefficient $\rho_{\mathrm{AE}} = 0.6$. Then, in exploitation we let the latter vary from 0 to 1 and we compute the KAP for $\mathrm{b} = 1, 2, 3, 4$ quantization bits. As shown in Fig. 6.8, each curve has a less steep behavior compared to Fig. 6.6. For example, with $\rho_{\mathrm{AE}} = 1$ and $\mathrm{b} = 2$ the value now settle to 0.7 instead of 1, a nice improvement with respect to the naive quantizer.

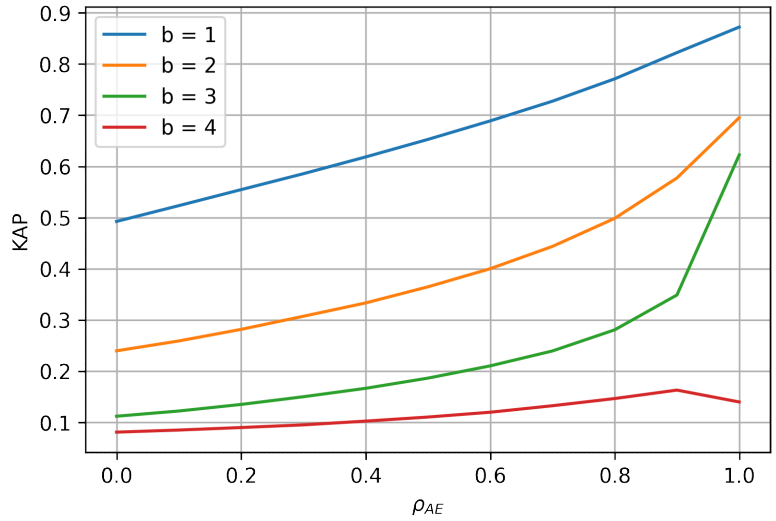


Figure 6.8: KAP between Alice and Eve for the developed quantizer with adversarial training.

Higher-dimensional input

We recall that we have generalized the developed joint quantizer by following an SVM-like approach to divide each user input space. Now it accepts an arbitrary number of features per user. We have trained such algorithm with the objective function (6.1).

Fig. 6.9 shows the mutual information between the initial keys of legitimate parties in function of ρ_{AB}.

6.1.2 Uniform joint quantizer

To have a baseline for comparison, Fig. 6.10 shows the SKR of the joint uniform quantizer. Also here the closed-form solutions for $\rho_{\mathrm{AB}} = 1$ and $\rho_{\mathrm{AE}} = 0$ coincide perfectly with the

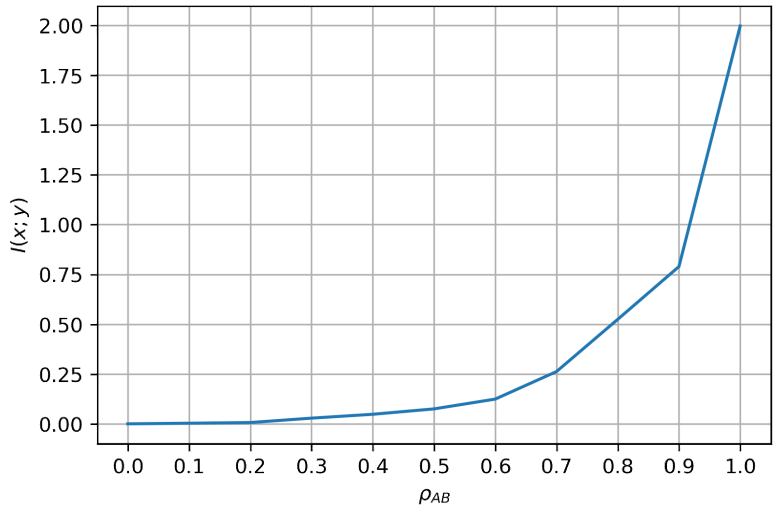


Figure 6.9: Mutual information between Alice and Bob for the higher-dimensional input version of the developed quantizer.

maximum of each curve. The SKR achieved by our algorithm is always far better with respect to the uniform joint quantizer.

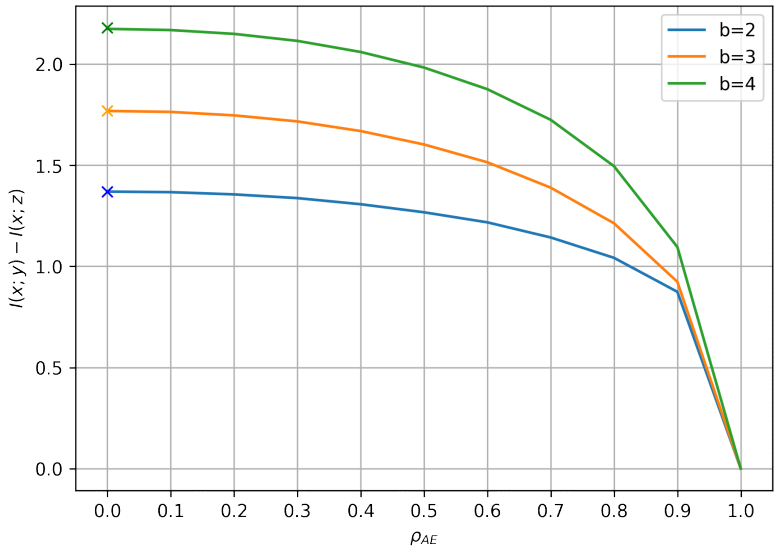


Figure 6.10: SKR in presence of an eavesdropper with the joint uniform quantizer.

6.1.3 Autoencoder with uniform quantizer

To evaluate the performance of such architecture, we will inspect the mutual information between Alice and Bob initial keys, namely, $\boldsymbol{z}_{\mathrm{A}}$ and $\boldsymbol{z}_{\mathrm{B}}$, the *Symbol Matching Rate*:

$$\mathrm{SMR}(\boldsymbol{z}_{\mathrm{A}}, \boldsymbol{z}_{\mathrm{B}}) = \frac{\sum_{i=1}^{|\mathcal{S}_{\mathrm{test}}|} \mathbb{1}\left\{\boldsymbol{z}_{\mathrm{A}}^{(i)} \neq \boldsymbol{z}_{\mathrm{B}}^{(i)}\right\}}{|\mathcal{S}_{\mathrm{test}}|}, \tag{6.7}$$

and the distribution of such initial keys.

Fig. 6.11 shows the distribution of the symbols constituting the legitimate parties initial keys, and we see that the AE strongly prefers two symbols and neglect the remaining two.

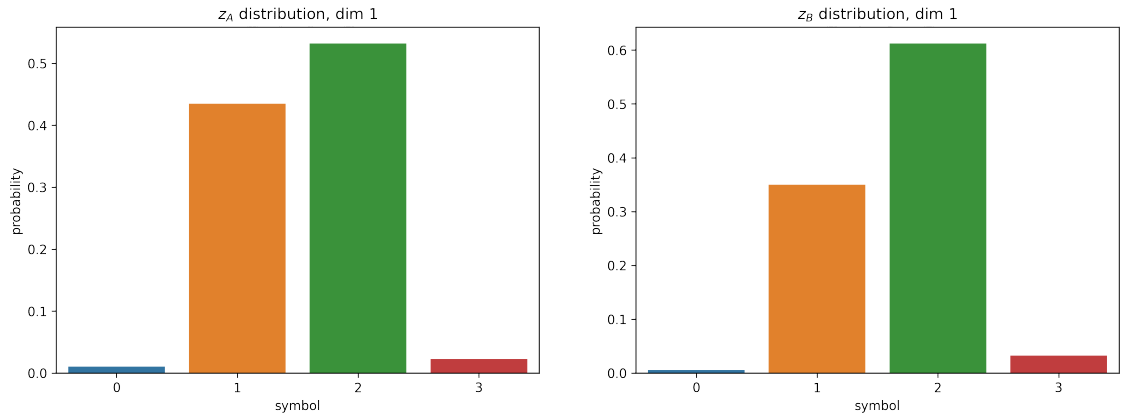


Figure 6.11: Symbol distribution of Alice and Bob's initial keys for the AE+UQ with $\mathrm{b} = 2$, Gaussian data set.

Fig. 6.12 shows the mutual information between Alice and Bob initial keys in function of the correlation ρ_{AB} between them. We can see that for more and more correlated measurements the mutual information tends to its maximum, which is $\mathrm{H}(\boldsymbol{z}) = 2, 4, 6$, and 8 bits/sample for $\mathrm{b} = 1, 2, 3$, and 4 quantization bits, respectively. This is double the normal maximum, and it is because we have a latent space of 2 neurons, each of them quantized with a given b so it is as we were using $2\mathrm{b}$ quantization bits.

Instead, Fig. 6.13 shows the SMR between Alice and Bob initial keys. Also here, the SMR tends to its maximum (i.e., 1) as the measurements of the two legitimate users become more correlated.

Lastly, in Fig. 6.14, where $\mathrm{I}(\boldsymbol{z}_{\mathrm{A}}; \boldsymbol{z}_{\mathrm{B}})$ is computed for $\rho_{\mathrm{AB}} = 0.9$, is shown the SKR in function of the correlation coefficient ρ_{AE} between Alice and Eve.

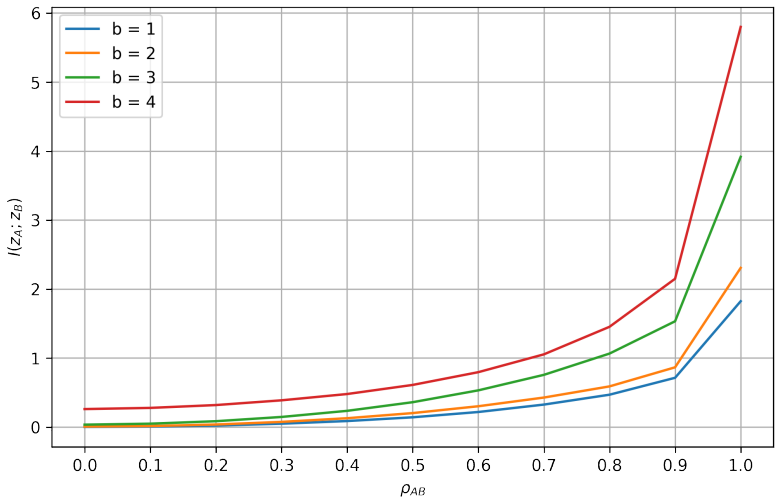


Figure 6.12: Mutual information between Alice and Bob initial keys for the AE+UQ, Gaussian data set.

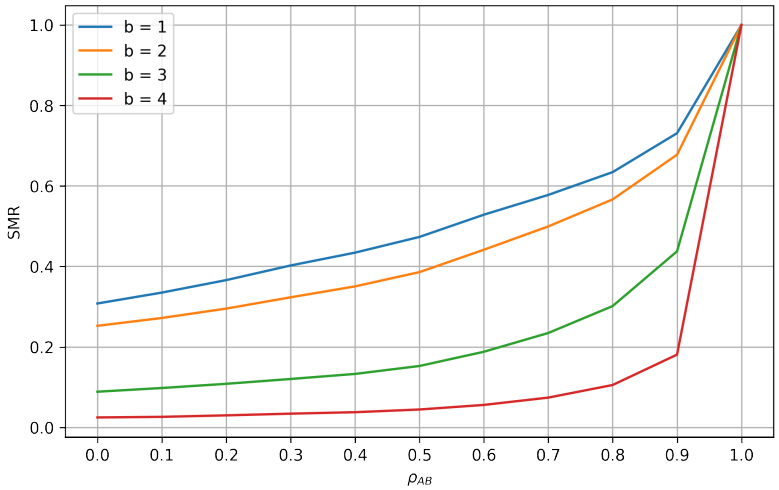


Figure 6.13: SMR between Alice and Bob for the AE+UQ, Gaussian data set.

For reference, we use PCA to perform dimensionality reduction from a four-dimensional input space to a compressed space of two dimensions. From Fig. 6.15, 6.16, and 6.17 we see that the mutual information, the SMR, and the SKR are almost the same to the ones achieved by the AE, respectively. This is because the AE uses linear activation functions, and thus it learns a mapping equivalent to the PCA [20]. However, we still prefer the AE as it can be trained in an adversarial manner to reduce the information related to Eve in the initial key of Alice and Bob.

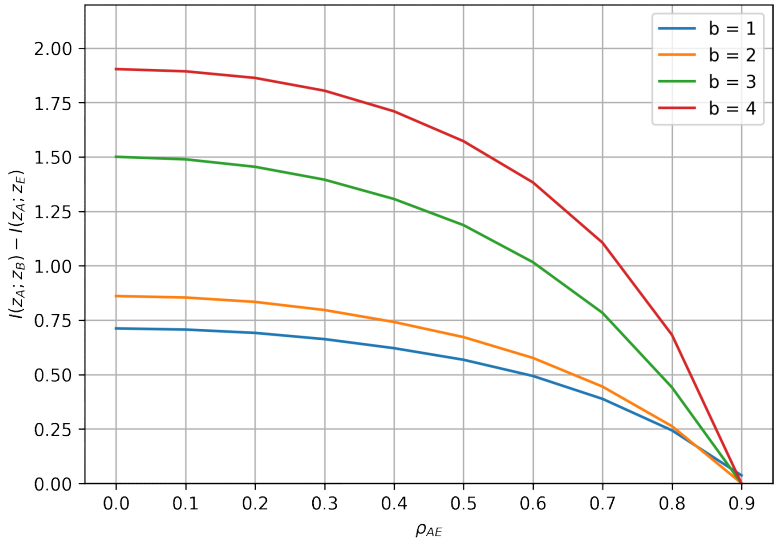


Figure 6.14: SKR between Alice and Bob for the AE+UQ, Gaussian data set.

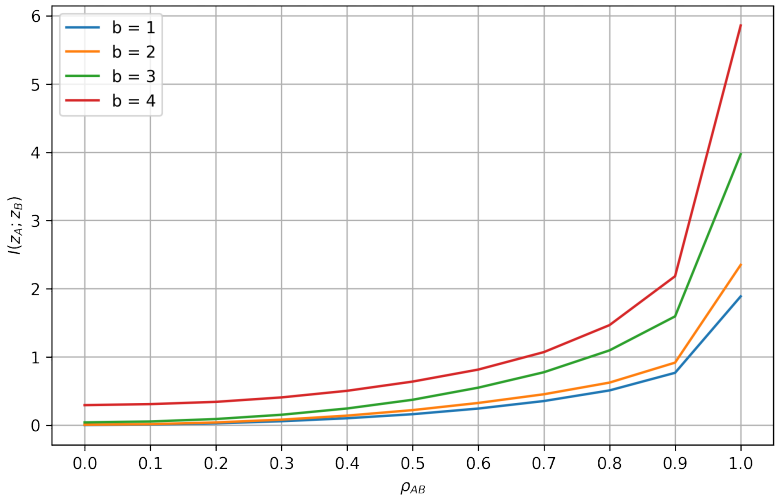


Figure 6.15: Mutual information between Alice and Bob initial keys with PCA, Gaussian data set.

6.1.4 Autoencoder with Differentiable Soft Quantization

In the training phase the transitions of the DSQ function change steepness, resembling more and more the hard uniform quantizer. Specifically, parameter κ is incremented by 10 every 10 epochs, so that the transitions between levels change steepness from $\kappa = 10$ to $\kappa = 250$.

In the exploitation phase, we compute the same metrics used in Section 6.1.3.

Fig. 6.18 shows the mutual information between Alice and Bob initial keys, while Fig. 6.19

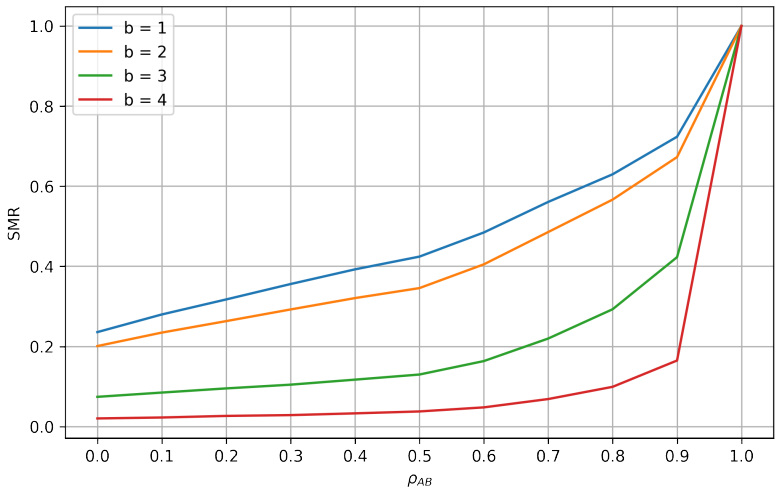


Figure 6.16: SMR between Alice and Bob with PCA, Gaussian data set.

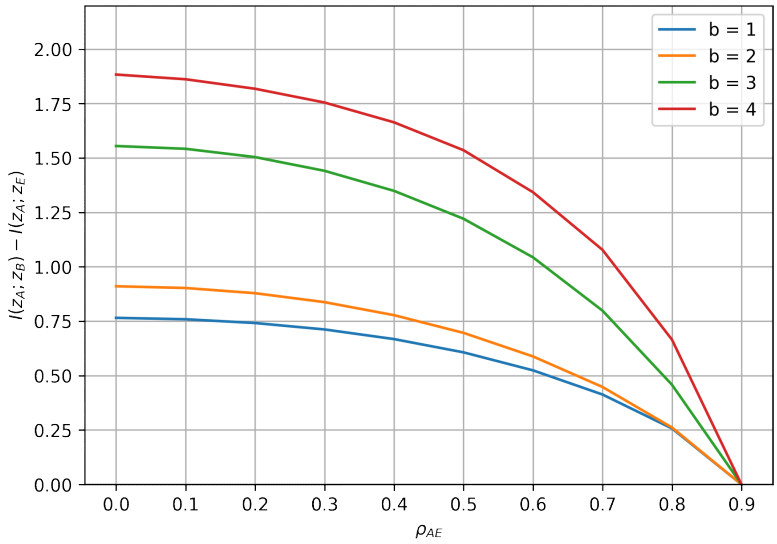


Figure 6.17: SKR between Alice and Bob with PCA, Gaussian data set.

shows the SMR, both in function of ρ_{AB}. The first is higher than the mutual information achieved by the AE+UQ for $\mathrm{b} = 1, 2$ while it remains almost the same for $\mathrm{b} = 3, 4$. Instead, the SMR is a bit lower with respect to the AE+UQ for low correlation values, whereas it remains almost the same for high correlation values.

However, this model still has the problem of yielding initial keys with poor randomness, and we are still doing nothing to protect from an eavesdropper. In fact, from Fig. 6.20 we see that

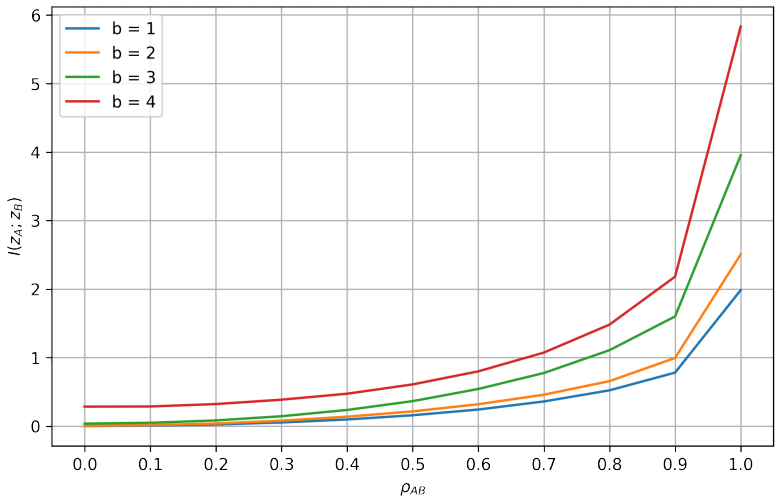


Figure 6.18: Mutual information between Alice and Bob for the AE+DSQ, Gaussian data set.

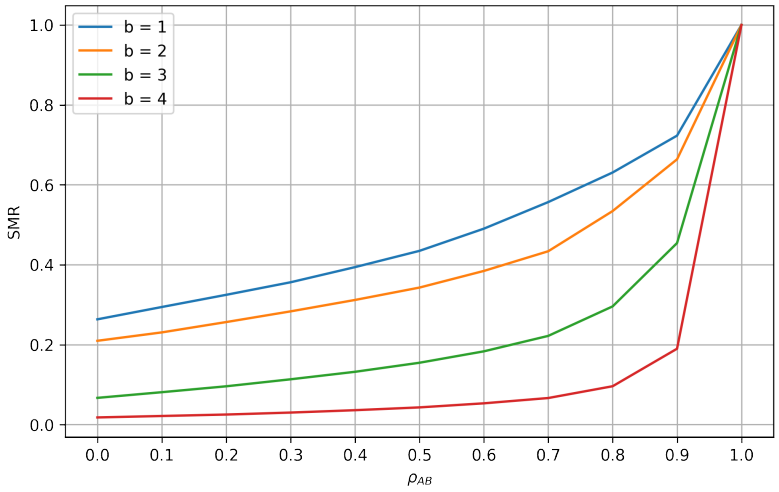


Figure 6.19: SMR between Alice and Bob for the AE+DSQ, Gaussian data set.

this architecture strongly prefers the symbols 1 and 2, and in Fig. 6.21 the SKR, despite being higher with respect to the AE+UQ, it still goes to zero when $\rho_{\mathrm{AE}} = \rho_{\mathrm{AB}}$.

6.1.5 MTAE

Here we use the MTAE architecture and training method described in [3]. However, the architecture is not the one illustrated in the paper as we operate with four channel features and not CIRs of size 300×2. The AE model is the same as the one detailed Section 5.3. Also the

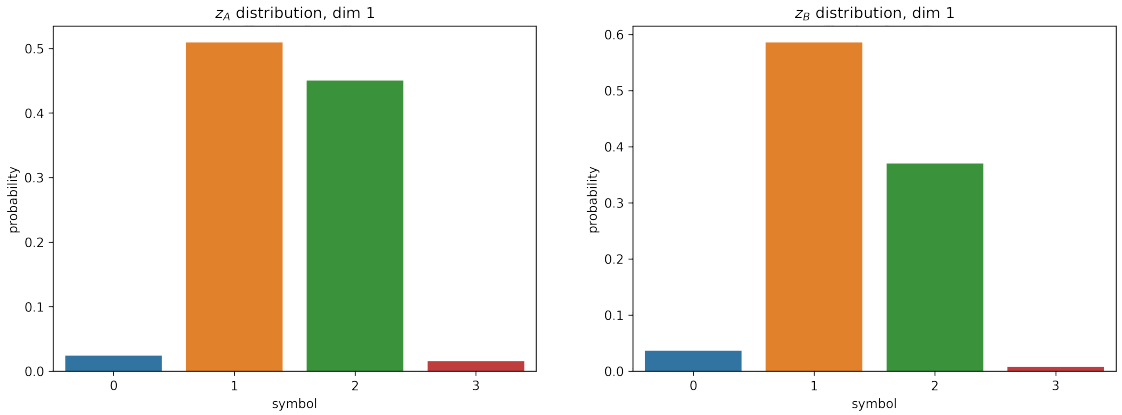


Figure 6.20: Symbol distribution of Alice and Bob's initial keys for the AE+DSQ, Gaussian data set.

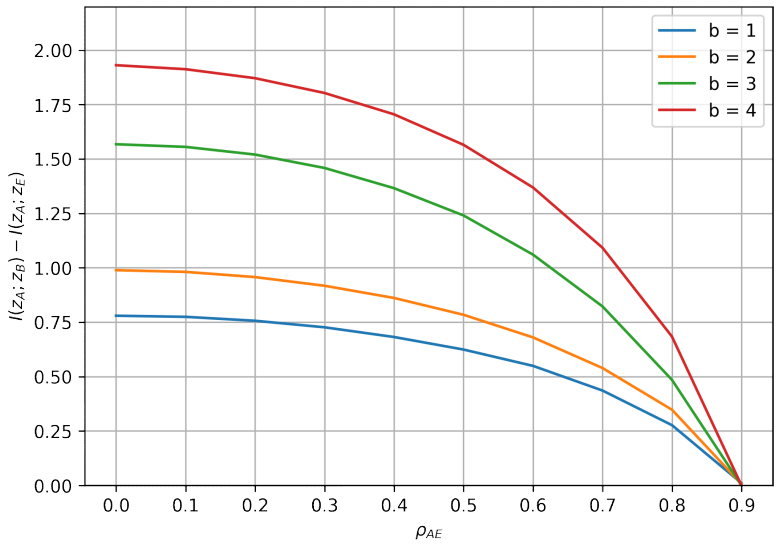


Figure 6.21: SKR between Alice and Bob for the AE+DSQ, Gaussian data set.

training differs a bit from the paper: we have trained the model for 250 epochs with the an exponential learning rate decay law

$$lr_{epoch} = \gamma lr_{epoch-1}, \tag{6.8}$$

with $\gamma = 0.97$.

To get the initial keys we can apply a uniform quantizer to the extracted features $\boldsymbol{z}_{\mathrm{A}}$, $\boldsymbol{z}_{\mathrm{B}}$, and $\boldsymbol{z}_{\mathrm{E}}$ in the exploitation phase, after we are done with the training in the real domain, or we can directly train the model with the DSQ layer embedded between encoder and decoder. We will

do both and compare the SKR of both methods with the SKR of the naive AE, trained in turn in the real domain and with DSQ.

As we can see from Fig. 6.22, the SKR starts from more or less the same values of the naive AE+UQ, but it does not go to zero anymore, and in particular all curves converges around 0.25 bits/sample. This means that the MTAE architecture is indeed capable of reducing the amount of information related to Eve contained in the reciprocal feature of Alice and Bob.

We then proceed by comparing the MTAE and the AE, both trained with DSQ, and we see that from Fig. 6.23 the SKR with 1-bit quantization start from a value almost the same to the SKR of the naive AE+DSQ, but it ends up in 0.25 bits/sample instead of collapsing to zero. Moreover, with 2-bit quantization the SKR starts lower than the naive AE+DSQ but it remains around 0.25 bits/sample instead of going to zero. Instead, for $\mathrm{b} = 3, 4$ the gain is significant for high ρ_{AE}, reaching values just above 0.5 and 0.75 bits/sample, respectively.

Finally, we also notice that the SKR of the MTAE+UQ architecture is slightly better for low ρ_{AE}, but for high values of ρ_{AE} the MTAE+DSQ architecture performs better.

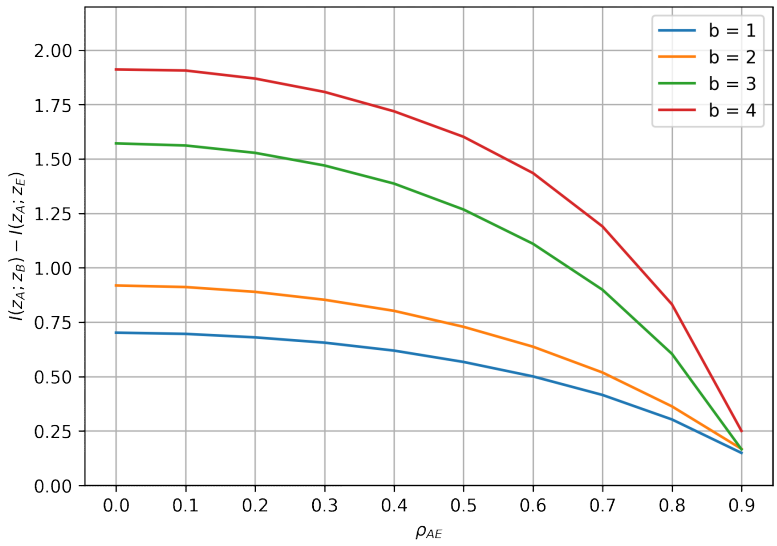


Figure 6.22: SKR between Alice and Eve for the MTAE+UQ architecture, Gaussian data set.

6.1.6 DAAE

Here we make use of the DAAE architecture and training method described in [4]. Also here, the encoder and the two decoders have been refactored, and have the same structure of the

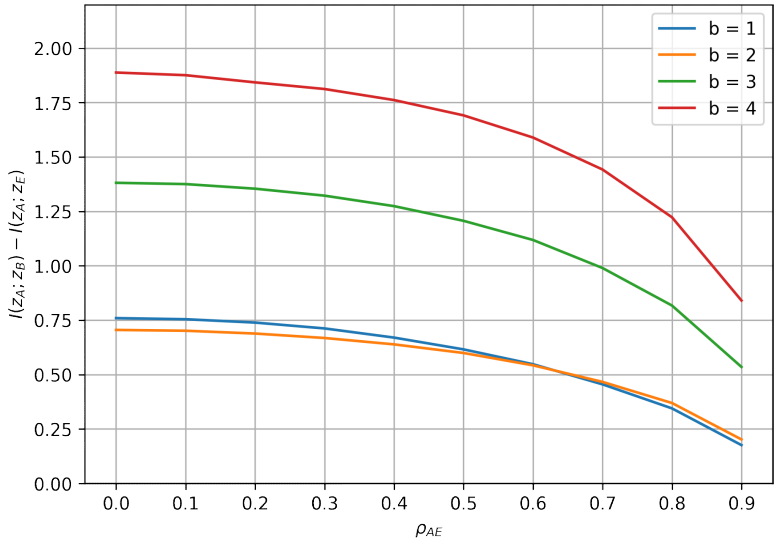


Figure 6.23: SKR between Alice and Eve for the MTAE+DSQ architecture, Gaussian data set.

encoder and decoder detailed in Section 5.3. Moreover, we use the same number of epochs and the same learning rate decay law of the MTAE.

Again, we can apply a uniform quantizer to the real-valued extracted features $\boldsymbol{z}_{\mathrm{A}}$, $\boldsymbol{z}_{\mathrm{B}}$, and $\boldsymbol{z}_{\mathrm{E}}$ in the exploitation phase, or we can directly train the model with the DSQ layer embedded between the encoder and the two decoders. We will compare the obtained SKR with the SKR of the MTAE architecture, for both cases.

From Fig. 6.24 it is clear that the DAAE+UQ outperforms the MTAE+UQ for all values of ρ_{AE} except $\rho_{\mathrm{AE}} = 0$, where the achieved SKR start from slightly less values. Now we can retain as much as ~ 0.6 bits/sample if we use 4-bit quantization.

From Fig. 6.25, for $\mathrm{b} = 1$ the DAAE+DSQ is worse with respect to the MTAE+DSQ but it is still better than the naive AE+DSQ. However, with $\mathrm{b} = 2$ the SKR is improved for all values of ρ_{AE}. Instead, with $\mathrm{b} = 3, 4$ the SKR is basically the same as the one achieved by the MTAE+DSQ.

6.1.7 AAE with uniform quantizer

Naive AAE

From Fig. 6.26 one can notice that the symbols contained in Alice's and Bob's primary keys are now almost equally likely, and the key randomness is hence higher with respect to the AE+UQ

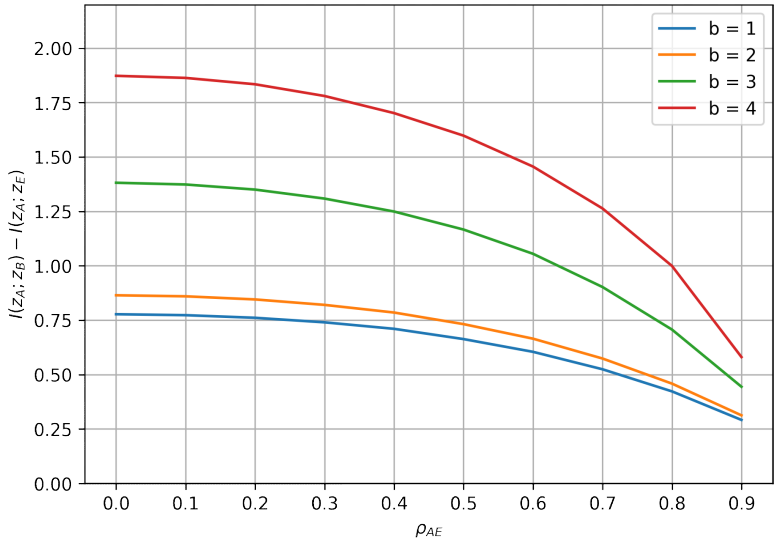


Figure 6.24: SKR between Alice and Eve for the DAAE+UQ architecture, Gaussian data set.

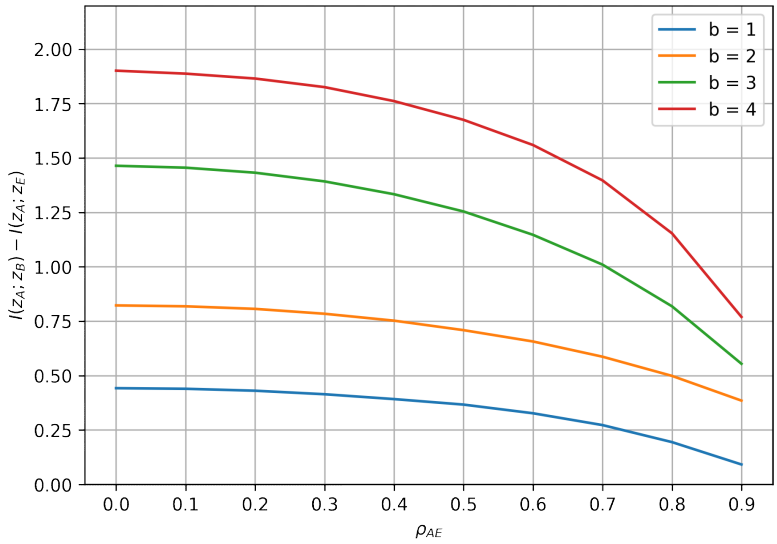


Figure 6.25: SKR between Alice and Eve for the DAAE+DSQ architecture, Gaussian data set.

and AE+DSQ.

The AAE architecture is capable of reaching almost the highest possible values of mutual information between Alice and Bob, while neither the AE+UQ nor the AE+DSQ are capable of reaching such values. In fact, from Fig. 6.27 for $\rho_{\mathrm{AB}} = 1$, the achieved mutual information is $\mathrm{I}(\boldsymbol{z}_{\mathrm{A}}, \boldsymbol{z}_{\mathrm{B}}) = \mathrm{H}(\boldsymbol{z}_{\mathrm{A}}) = \mathrm{H}(\boldsymbol{z}_{\mathrm{B}}) = \{2, 4, 6, 8\}$ for $\mathrm{b} = 1, 2, 3, 4$, respectively. As explained

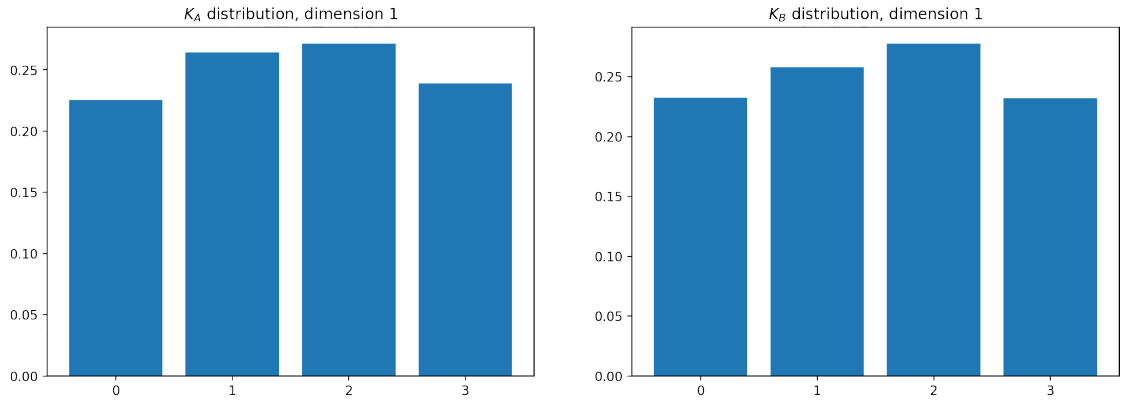


Figure 6.26: Primary key distributions for Alice (left plot) and Bob (right plot), Gaussian data set.

before, this is double the canonical maximum, and it is because we have a latent space of 2 neurons, each of them quantized with a given b so it is as we were using $2b$ quantization bits. The mutual information gain comes with the cost of a lower SMR. As can be seen in Fig. 6.28,

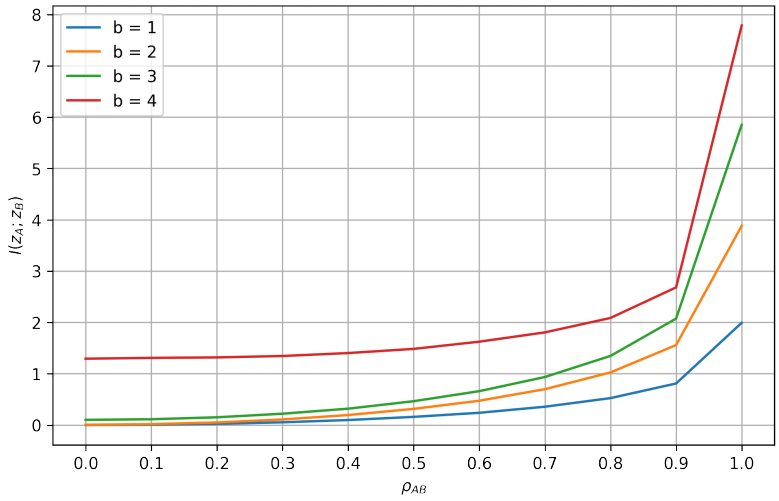


Figure 6.27: Mutual information between Alice and Bob for the AAE architecture, Gaussian data set.

apart from the case $b = 1$, for which the curve behaves more or less the same with respect to the AE+UQ, the other three curves are way worse. Instead, the SMR of the AE+DSQ is always higher than the AAE in all cases. Ultimately, from Fig. 6.29 we see that the SKR is in general greatly improved with respect to the AE. However for $b = 4$ we notice that the SKR for $\rho_{\mathrm{AE}} = 0$ starts from a too low value, whereas it should start around 3 bits/sample. This strange behavior can be explained by the high amount of mutual information that the architecture is

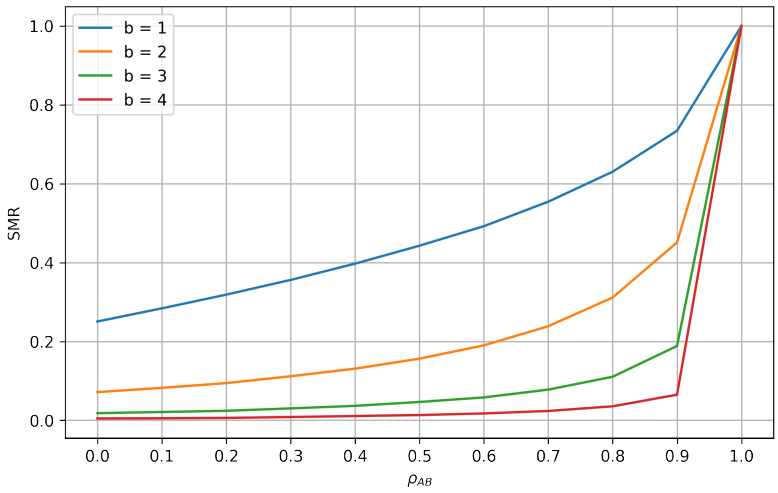


Figure 6.28: SMR between Alice and Bob for the AAE architecture, Gaussian data set.

providing even at a correlation factor of zero when $\mathrm{b} = 4$, as one can see from Fig. 6.27. So even if Eve is poorly correlated, she is capable of producing features through the encoder that contains a lot of information regarding Alice or Bob. Aside from this, since we have not done anything to protect ourselves from an eavesdropper when this is less than half a wavelength far from either of legitimate parties, the SKR is bound to go to zero for $\rho_{\mathrm{AB}} = \rho_{\mathrm{AE}}$.

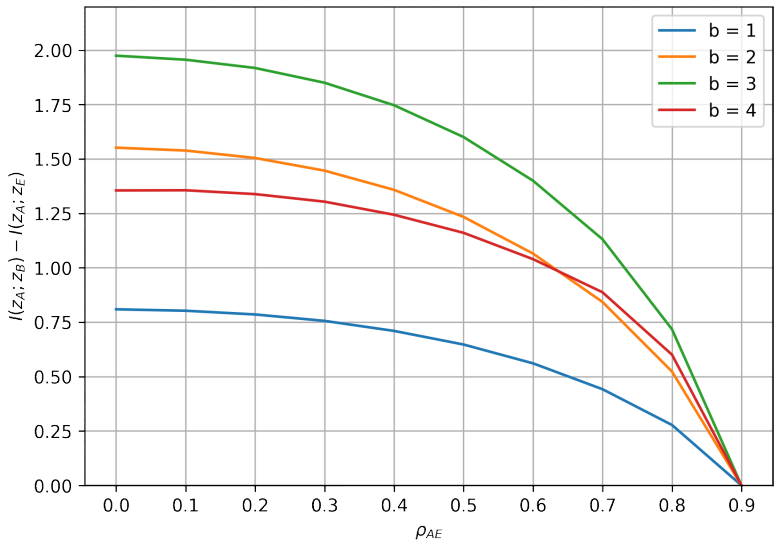


Figure 6.29: SKR for the naive AAE architecture, Gaussian data set.

To wrap up, the AAE architecture performs a lot better in terms of mutual information

between the legitimate parties, but the SKR has the same behavior of the AE architectures described in Section 5.4 and in Section 5.5. We indeed need to train the AAE to take into account also the eavesdropper presence.

Eavesdropper-aware AAE

We plot again the SKR in function of ρ_{AE}. From Fig. 6.30 we see that there is a small gain for high values of ρ_{AE} and the SKR does not collapse to zero when $\rho_{\mathrm{AB}} = \rho_{\mathrm{AE}} = 0.9$.

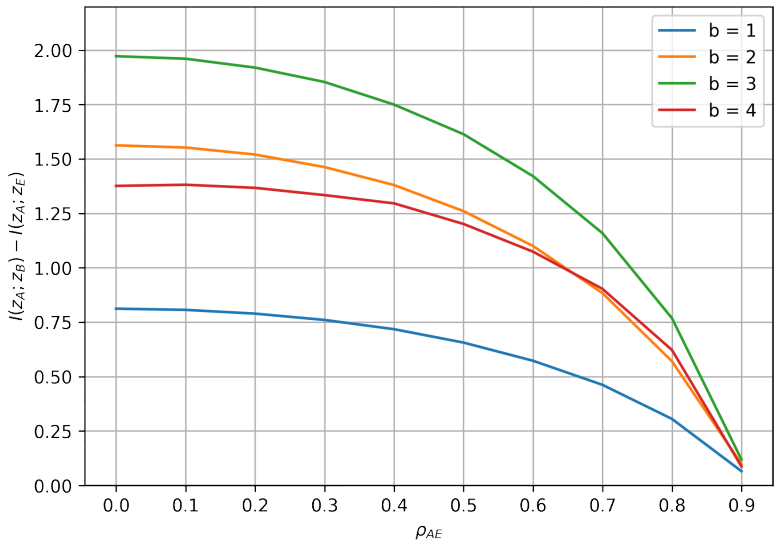


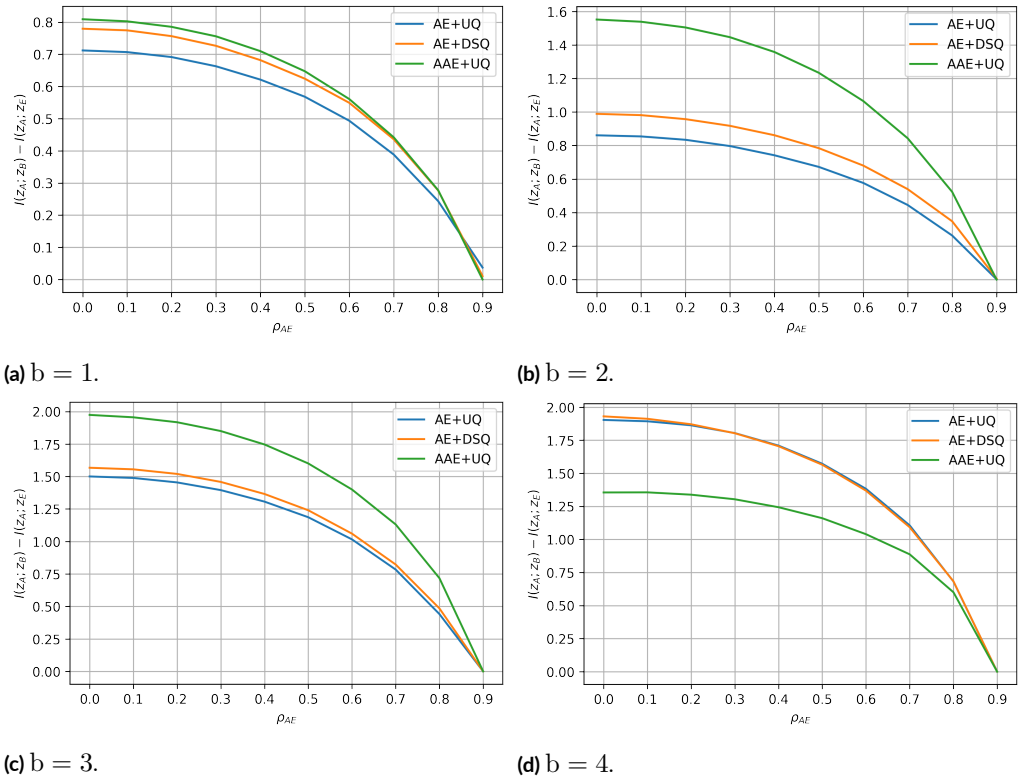
Figure 6.30: SKR for the AAE architecture with eavesdropper-aware training, Gaussian data set.

6.1.8 Comparison between NN architectures

Lastly, to better realize the differences between the achieved SKRs of all the presented architectures, we plot the SKR curves of each model for a particular b in the same graph. This is repeated for each $\mathrm{b} \in \{1, 2, 3, 4\}$.

In particular, Fig. 6.31 shows the SKR curves of the AE trained in the real domain, the AE trained with DSQ, and the AAE for 1-, 2-, 3-, and 4-bit quantization. Apart from the case $\mathrm{b} = 4$ (the reason of which was already explained in Section 6.1.7), we confirm that the proposed AAE architecture is better than the AE.

Instead, Fig. 6.32 shows the SKR curves of MTAE, DAAE (both trained in the ral domain and with the DSQ), and the AAE with adversarial-aware training. For $\mathrm{b} = 1, 2, 3$ the SKR of

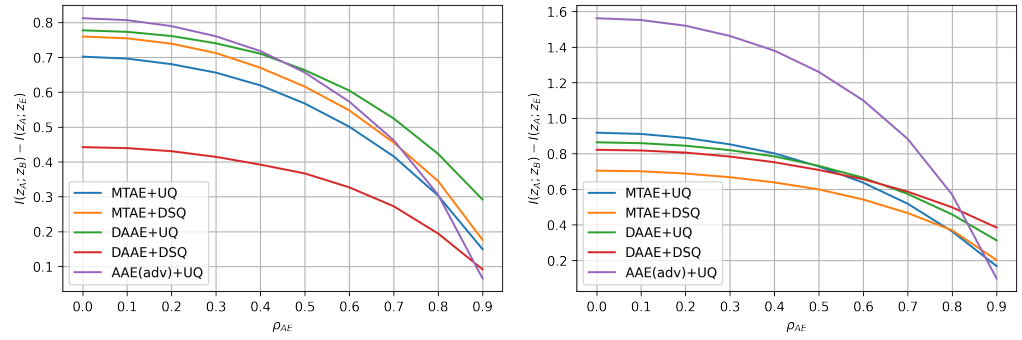


(a) $\mathrm{b} = 1$. **(b)** $\mathrm{b} = 2$.

(c) $\mathrm{b} = 3$. **(d)** $\mathrm{b} = 4$.

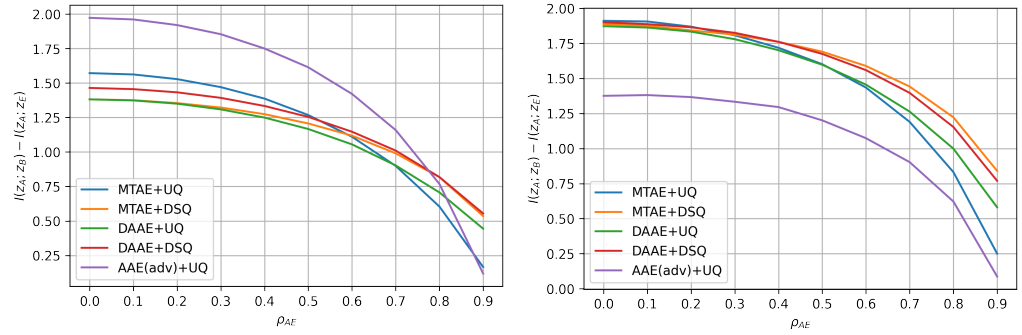
Figure 6.31: Comparison of SKR curves of AE+UQ, AE+DSQ, and AAE+UQ, Gaussian data set.

the AAE architecture is better than the others until $\rho_{\mathrm{AE}} = 0.8$, while beyond this value the other architectures retain higher rates. Again, the case $\mathrm{b} = 4$ is problematic as before.



(a) $\mathrm{b} = 1$.

(b) $\mathrm{b} = 2$.



(c) $\mathrm{b} = 3$.

(d) $\mathrm{b} = 4$.

Figure 6.32: Comparison of SKR curves of MTAE+UQ, MTAE+DSQ, DAAE+UQ, DAAE+DSQ, and AAE(adv)+UQ, Gaussian data set.

6.2 Underwater data set

6.2.1 Developed joint quantizer

In this section we will just repeat some of the previous algorithms described in Section 6.1.1 with with the underwater data set detailed in Section 3.3.. In particular, these are stored in two `csv` files: one of these represents the channel features of Eve, and call it $\boldsymbol{z}$; from the remaining one two data sets are generated by adding a variable amount of white noise and assigned to Alice and Bob, respectively, and call them $\boldsymbol{x}$ and $\boldsymbol{y}$. Specifically, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are the two original data sets, we normalize them in order to have each feature to be zero mean and unit variance and then we do:

$$\begin{aligned} \boldsymbol{x} &= \mathcal{S}_1 + \boldsymbol{w}_x \\ \boldsymbol{y} &= \mathcal{S}_1 + \boldsymbol{w}_b \\ \boldsymbol{z} &= \mathcal{S}_2 \end{aligned} \tag{6.9}$$

where $\boldsymbol{w}_x^{(i)}, \boldsymbol{w}_y^{(i)} \sim \mathcal{N}(\mu = 0, \sigma^2 = \frac{1}{\mathrm{SNR}})$. This operation is repeated for each correlation value between the two original `csv` files.

First we discuss the results for the two-dimensional quantizer. Fig. 6.33 shows the KAP in function of the SNR (expressed in decibels) for $\mathrm{b} = 2, 3, 4$ quantization bits, i.e., for $\mathrm{R} = 2^{\mathrm{b}} = 4, 8, 16$ regions. As expected, increasing the number of quantization bits leads to a lower KAP, due to the nature of the developed quantizer: with many rectangles to divide the data into, the algorithm is forced to leave many points outside them, meaning that Alice and Bob measurements are less likely to fall into the same regions if they are weakly correlated. Here we have used only the first feature, as the initial version algorithm only supports one-dimensional data as the input for each user. Then, we proceed to investigate the performance with the presence of an eavesdropper. By varying the correlation factor $\rho_{\mathrm{AE}} \in [0, 1]$ and having Eve applying the same strategy described in Section 4.1.2, we plot the SKR for each of the four channel features in Fig. 6.34. From this figure, the fourth feature seems to best to choose, since the SKR starts from a higher point with respect to the other but also the rate of descent is lower than the others. However, if we compute the KAP between Alice and Eve for each channel feature as a function of ρ_{AE}, for the first and second features the probability of Eve finding the same key of Alice grows slower than for the third and fourth features. Thus from this points of view focusing on extracting the key on the first two features leads to a less consistent initial key with respect to Eve.

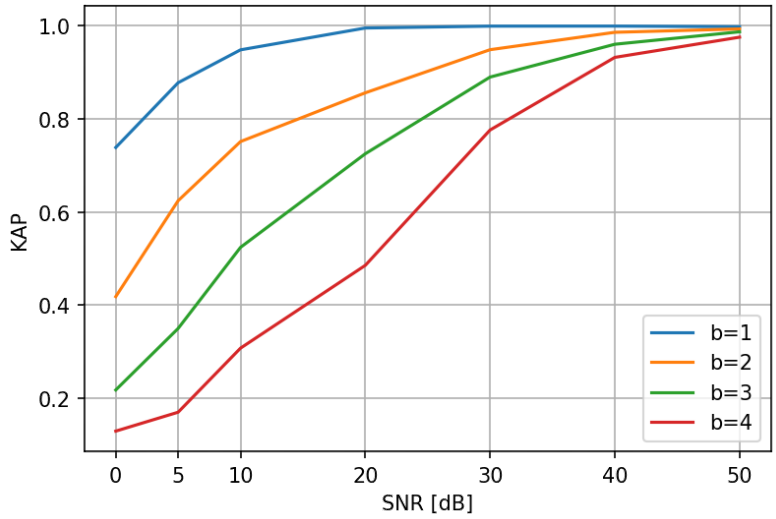
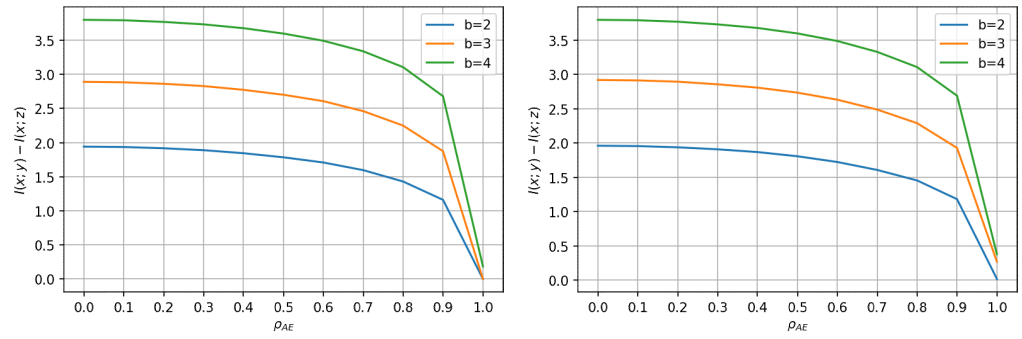
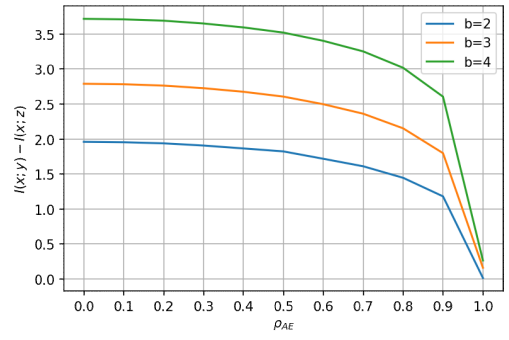


Figure 6.33: KAP between Alice and Bob in function of SNR.

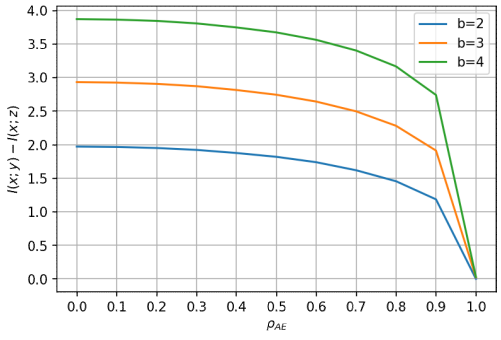


(a) First feature.

(b) Second feature.



(c) Third feature.



(d) Fourth feature.

Figure 6.34: SKR of the developed joint quantizer in function of ρ_{AE} for each channel feature of the underwater data set.

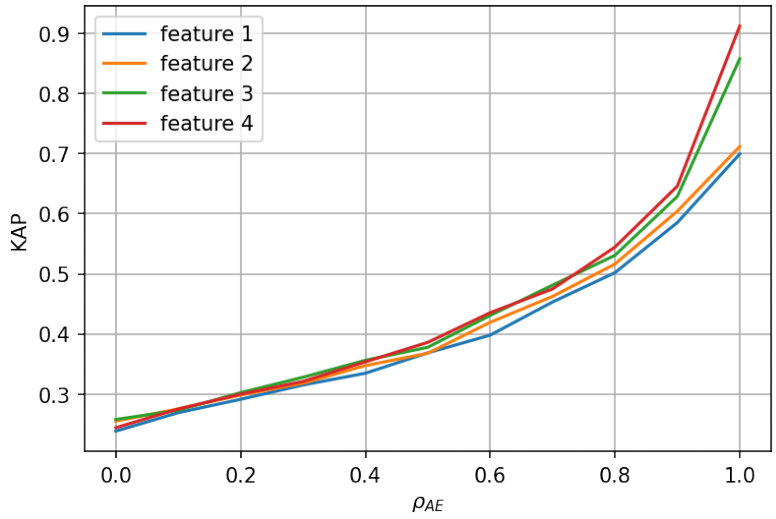


Figure 6.35: KAP vs ρ_{AE} with $\mathrm{R} = 4$.

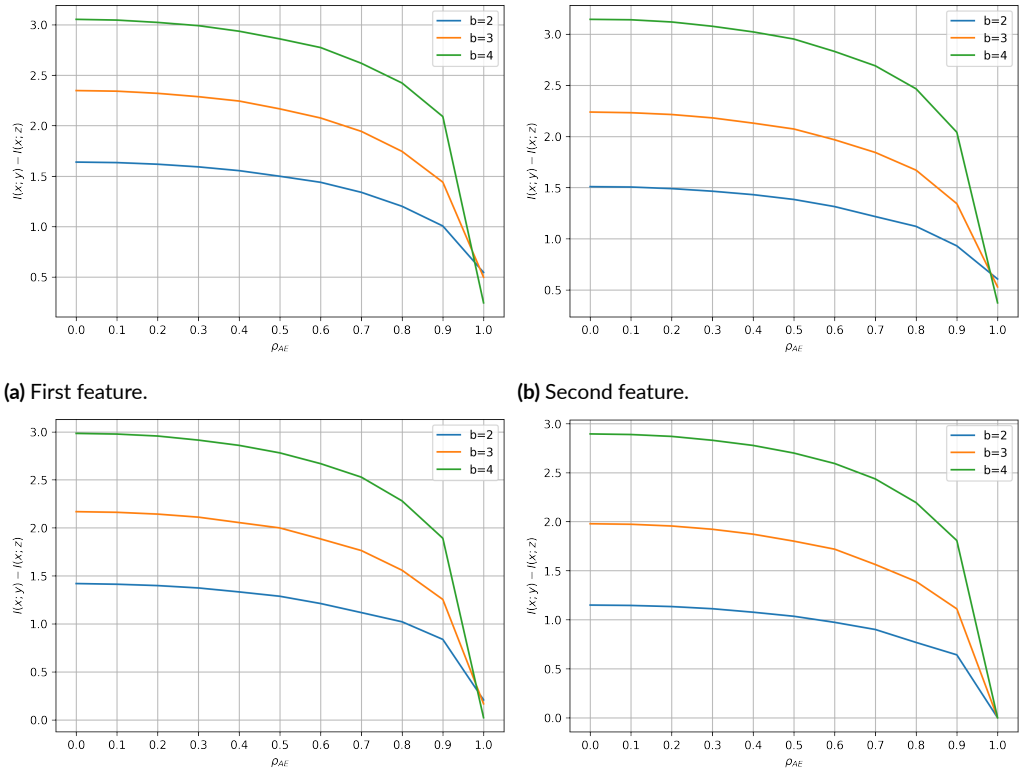
6.2.2 Uniform joint quantizer

For reference, we plot the SKR of the joint uniform quantizer for each channel feature of the underwater data set in Fig. 6.36. We see that the achieved SKR of our joint quantizer is better than the first. For the first and second channel feature we notice that the SKR remains quite high for $\rho_{\mathrm{AE}} = 1$, around 0.5, and this confirms the fact that our developed algorithm can better extract all the possible information. That is, for Eve the best situation is when she can adopt the developed joint quantizer instead of the joint uniform quantizer, which does not allow her to completely exploit the correlation with respect to Alice.

6.2.3 AE with uniform quantizer

From now on, let the data series for Alice, Bob, and Eve be $\boldsymbol{x}_{\mathrm{A}}$, $\boldsymbol{x}_{\mathrm{B}}$, and $\boldsymbol{x}_{\mathrm{E}}$, respectively.

As for the Gaussian data set, from Fig. 6.37 we see that the initial key randomness is quite poor, that is, the two "inner" symbols are more probable than the other two, as expected. Instead, the mutual information between legitimate parties, shown in Fig. 6.38 reaches higher values for $\mathrm{b} = 2, 3, 4$. From Fig. 6.39 we verify that the SMR between Alice and Bob's initial keys has the intended behavior, i.e, it tends to 1 for high SNR values. The SKR, shown in Fig. 6.40, computed on the underwater data set is better than the SKR for the Gaussian data set, and in addition it does not collapse to zero when $\rho_{\mathrm{AE}} = \rho_{\mathrm{AB}}$.



(a) First feature.

(b) Second feature.

(c) Third feature.

(d) Fourth feature.

Figure 6.36: SKR of the joint uniform quantizer in function of ρ_{AE} for each channel feature of the underwater data set.

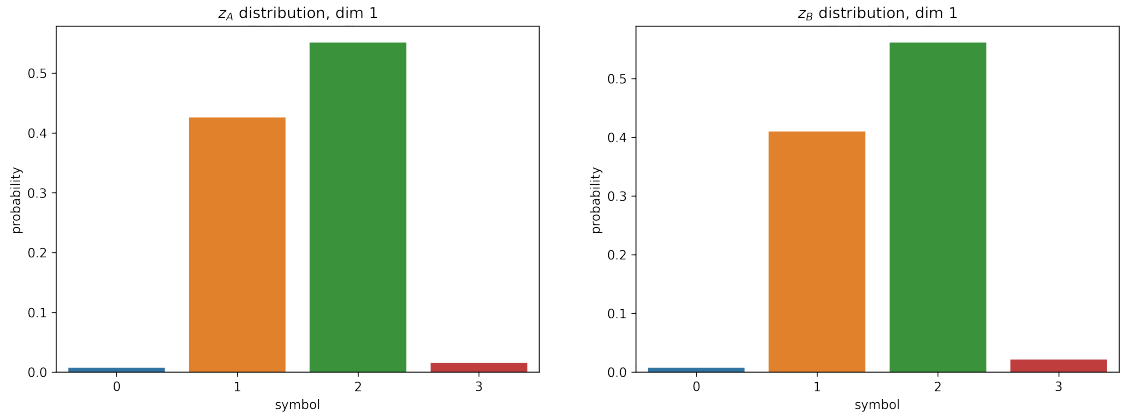


Figure 6.37: Symbol distribution of Alice and Bob's initial keys for the AE with $\mathrm{b} = 2$, underwater data set.

6.2.4 AE WITH DSQ

Also for the underwater data set, the mutual information between Alice and Bob for the AE trained with DSQ, shown in Fig. 6.42, is worse than the AE trained in the real domain. In

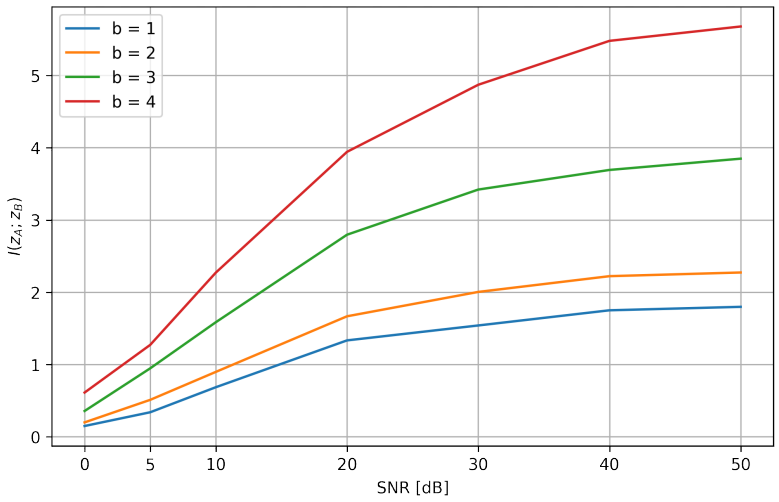


Figure 6.38: Mutual information between Alice and Bob initial keys for the AE+UQ, underwater data set.

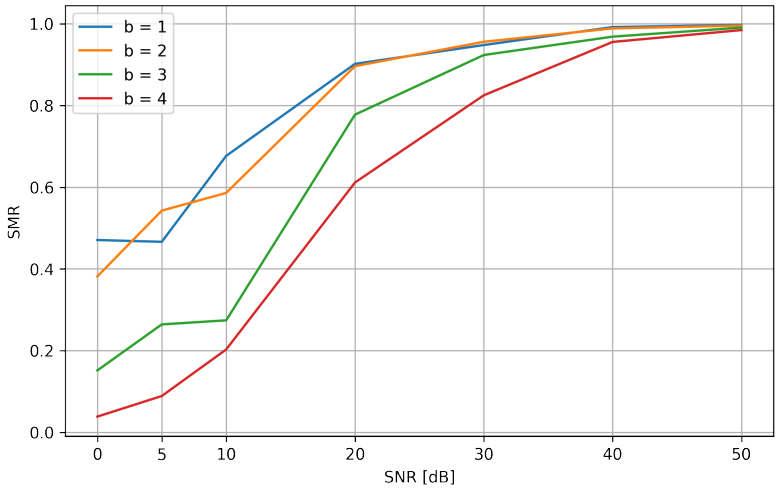


Figure 6.39: SMR between Alice and Bob for the AE+UQ, underwater data set.

particular, for $\mathrm{b} = 1$ we have half the achieved mutual information with respect to the latter, while in the other cases the values are only slightly less. From Fig. 6.43 we see that the SMR between Alice and Bob has the expected trend, going to 1 for high values of SNR. The SKR for the AE trained with DSQ, shown in Fig. 6.44, starts from lower values with respect to the AE trained in the real domain. However, for high values of ρ_{AE} it retains higher rates for $\mathrm{b} = 3, 4$ with respect to the latter.

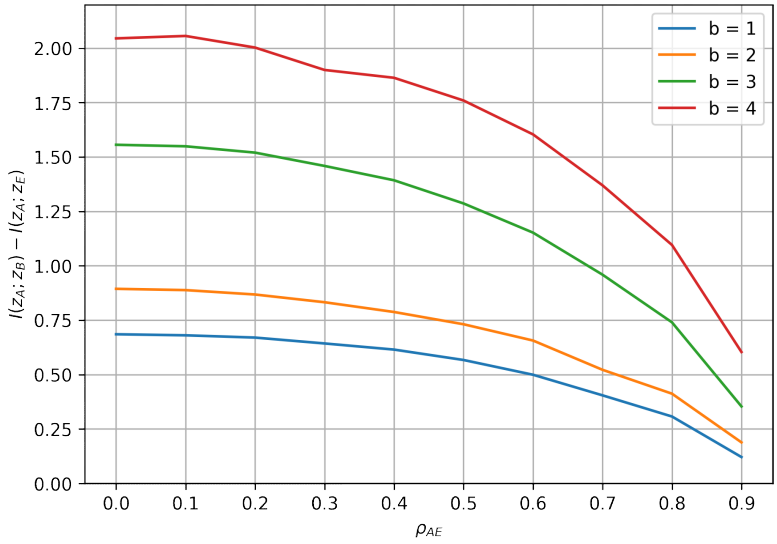


Figure 6.40: SKR between Alice and Bob for the AE+UQ, underwater data set.

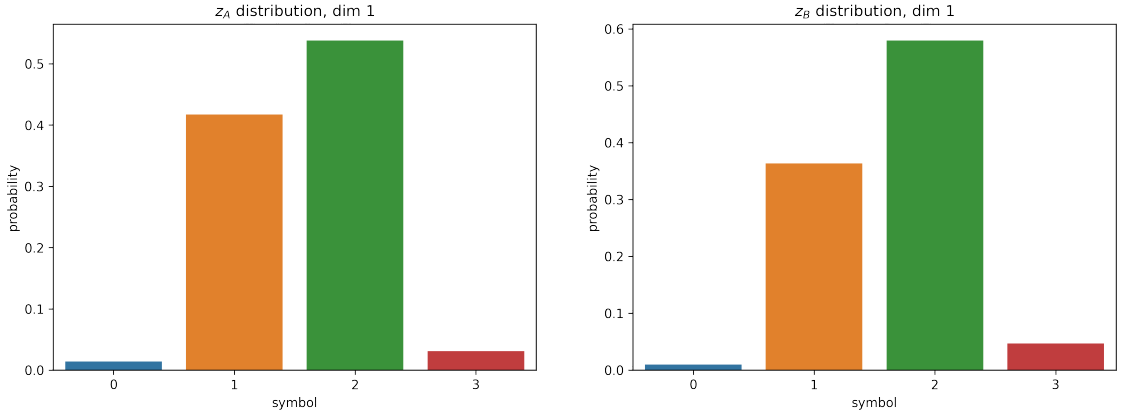


Figure 6.41: Symbol distribution of Alice and Bob's initial keys for the AE+DSQ, underwater data set.

6.2.5 MTAE

Here we first compare the MTAE trained in the real domain and with an UQ applied on top of the encoder in the exploitation phase with the AE trained in the real domain, then we compare the MTAE+DSQ and the AE+DSQ architecture, and finally we will discuss whether it is better the MTAE+UQ or the MTAE+DSQ on the basis of the outcomes.

From Fig. 6.45 we do not see any improvement with respect to the AE+UQ, while from Fig. 6.46 we realize the MTAE+DSQ is even worse than the AE+DSQ for $\mathrm{b} = 1, 2$. Nonetheless, the MTAE trained in the real domain performs better than the MTAE trained with the

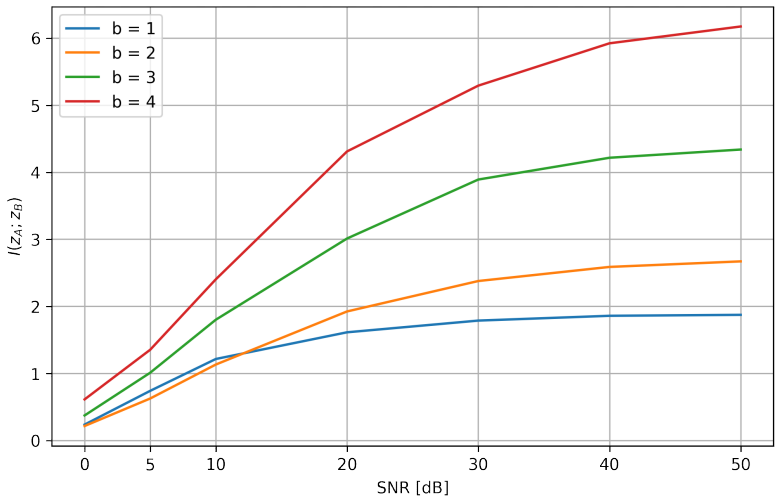


Figure 6.42: Mutual information between Alice and Bob initial keys for the AE+DSQ, underwater data set.

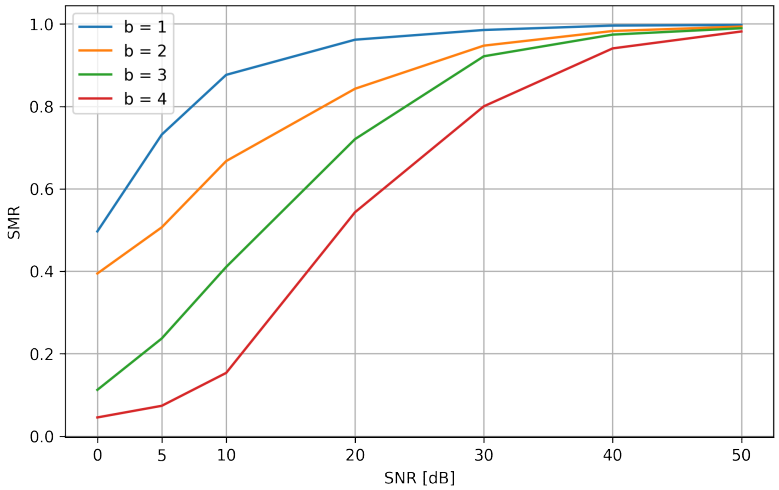


Figure 6.43: SMR between Alice and Bob for the AE+DSQ, underwater data set.

DSQ.

6.2.6 DAAE

Here we first compare the DAAE trained in the real domain and with an UQ applied on top the encoder in the exploitation phase with the AE trained in the real domain, then we compare the DAAE+DSQ and the AE+DSQ architecture, and finally we will discuss whether it is better the DAAE+UQ or the DAAE+DSQ on the basis of the outcomes.

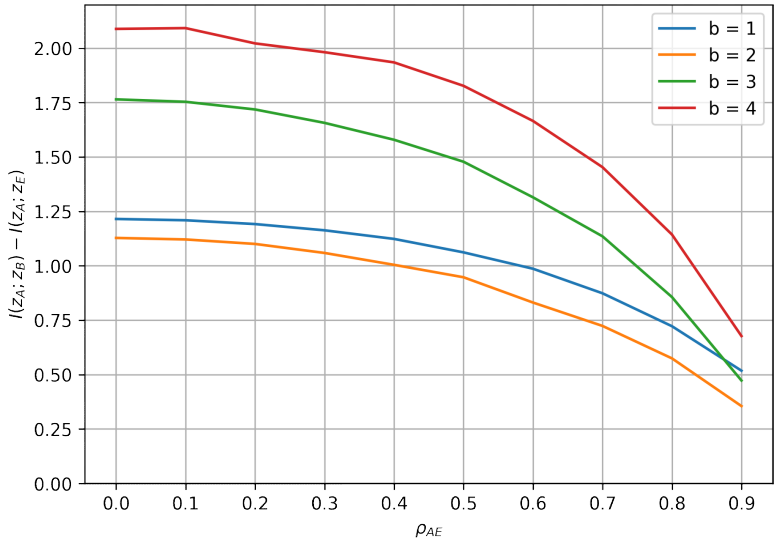


Figure 6.44: SKR between Alice and Bob for the AE+DSQ, underwater data set.

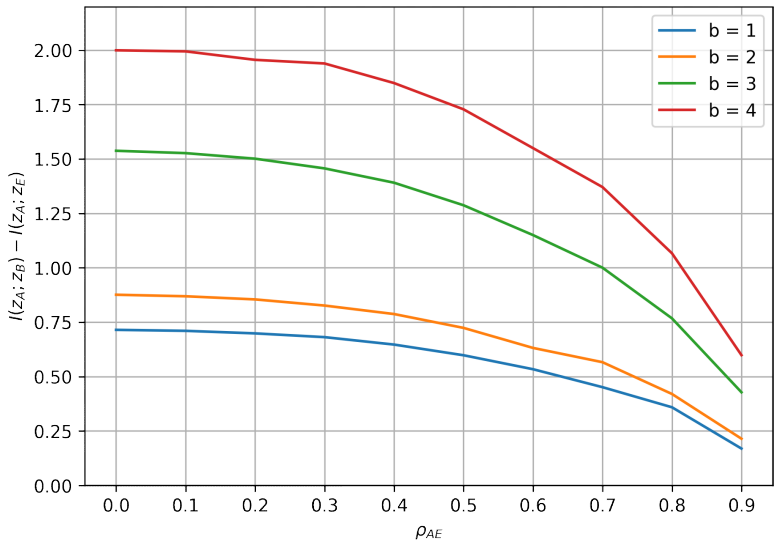


Figure 6.45: SKR between Alice and Bob for the MTAE+UQ, underwater data set.

From Fig. 6.47 we see that the SKR achieved by DAAE+UQ is always better than the AE+UQ for $\mathrm{b} = 1, 2, 3$, while with $\mathrm{b} = 4$ there is no noticeable improvement. Instead, from Fig. 6.48 we see that the DAAE+DSQ is worse than the AE+DSQ for every value of b and rho_{AE}. The DAAE+UQ is indeed better than the DAAE+DSQ with underwater measurements. Moreover, the DAAE+UQ always outperforms the MTAE+UQ, whereas the

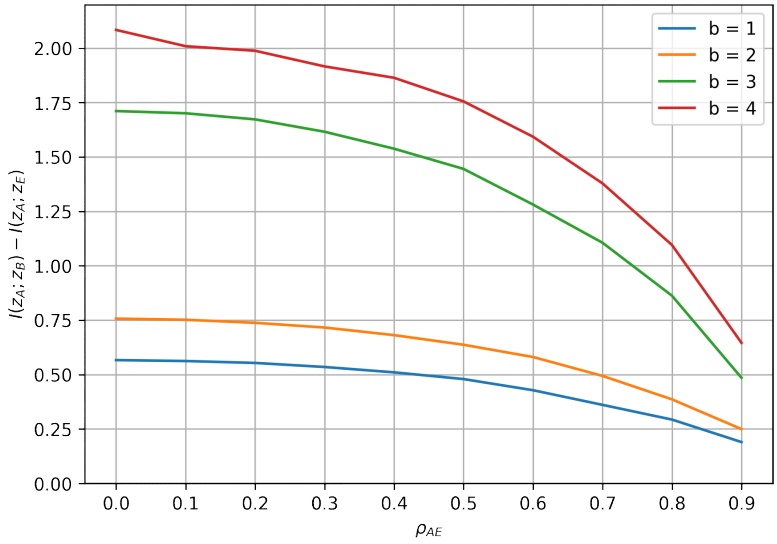


Figure 6.46: SKR between Alice and Bob for the MTAE+DSQ, underwater data set.

DAAE+DSQ is better than the MTAE+DSQ for $\mathrm{b} = 1, 2$, and more or less has the same performance for $\mathrm{b} = 3, 4$.

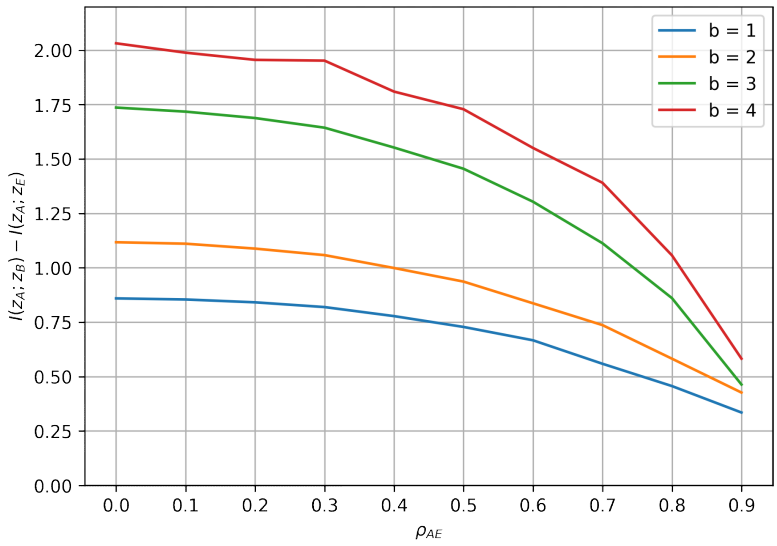


Figure 6.47: SKR between Alice and Bob for the DAAE+UQ, underwater data set.

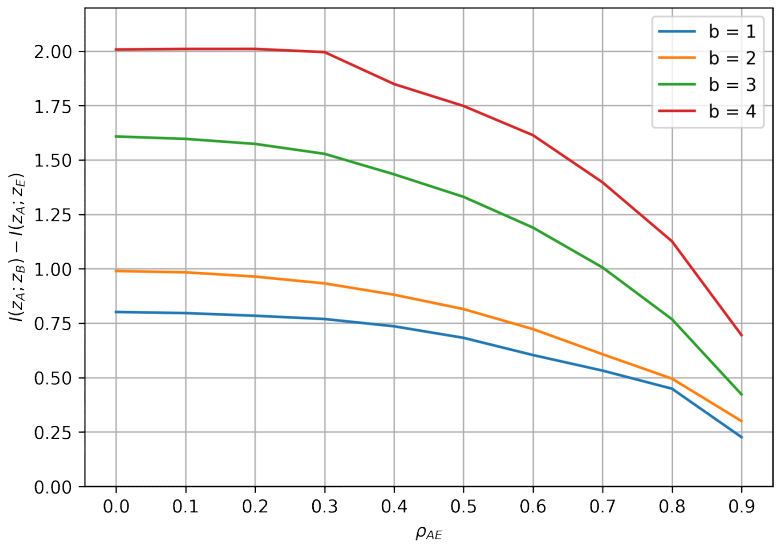


Figure 6.48: SKR between Alice and Bob for the DAAE+DSQ, underwater data set.

6.2.7 AAE

We train the same model described in Section 5.5 with the underwater data set and report here the results.

We recall that the objective, besides learning a reciprocal latent space z, is to have this to be distributed uniformly. From Fig. 6.49 we see that the distribution of the produced latent space is indeed resembling a uniform distribution. Thanks to this, when the uniform quantizer is applied in the exploitation phase to output the initial key, the symbols composing it will be more equally likely, as in Fig. 6.50. The key randomness is slightly worse than the one obtained with the Gaussian data set. Due to this, also the mutual information between Alice and Bob is lower than the achievable maximum with respect to the AAE trained with the Gaussian data set, as evidenced by Fig. 6.51. Despite this architecture not being trained in an adversarial-aware manner, the SKR does not go to zero when $\rho_{\mathrm{AE}} = \rho_{\mathrm{AB}}$, as shown in Fig. 6.52. Moreover, we expect the red curve (representing 4-bit quantization per neuron) to start at a higher value with respect to the green curve (representing 3-bit quantization per neuron). As for the Gaussian data set, this should be because of the high mutual information that the AAE is capable of extracting for $\mathrm{b} = 4$ even at very low SNR values. If we train this model in an adversarial-aware manner, i.e., with the MTAE paradigm, we obtain the SKR in Fig. 6.53. If for $\mathrm{b} = 2$ and $\mathrm{b} = 3$ there is no substantial improvement, we see that the red and blue curve are now better for any value ρ_{AE} with respect to the naive AAE.

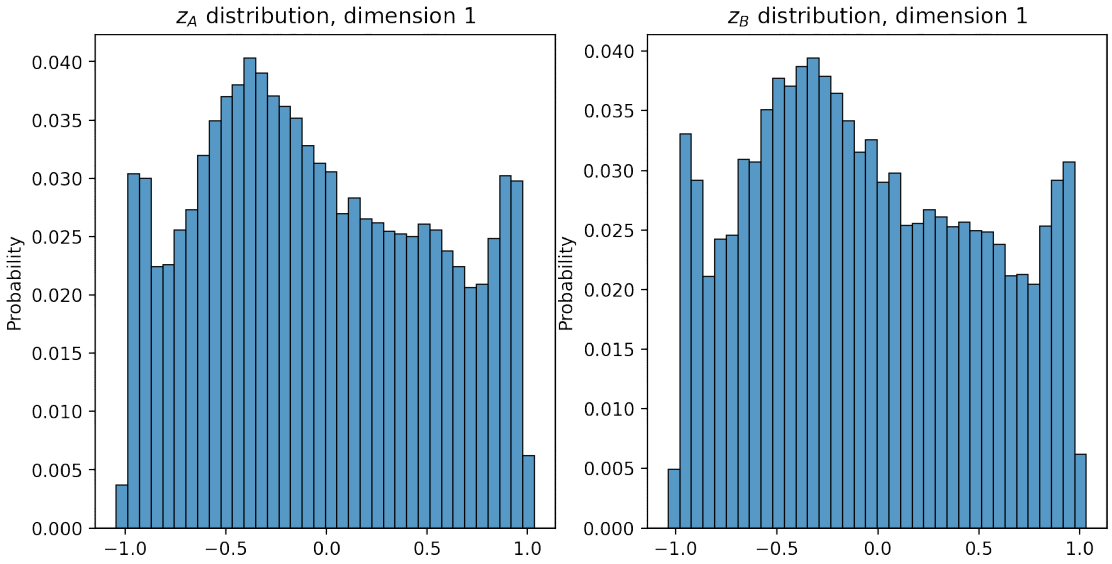


Figure 6.49: Latent space distributions for Alice's (left plot) and Bob's (right plot) primary keys, underwater data set.

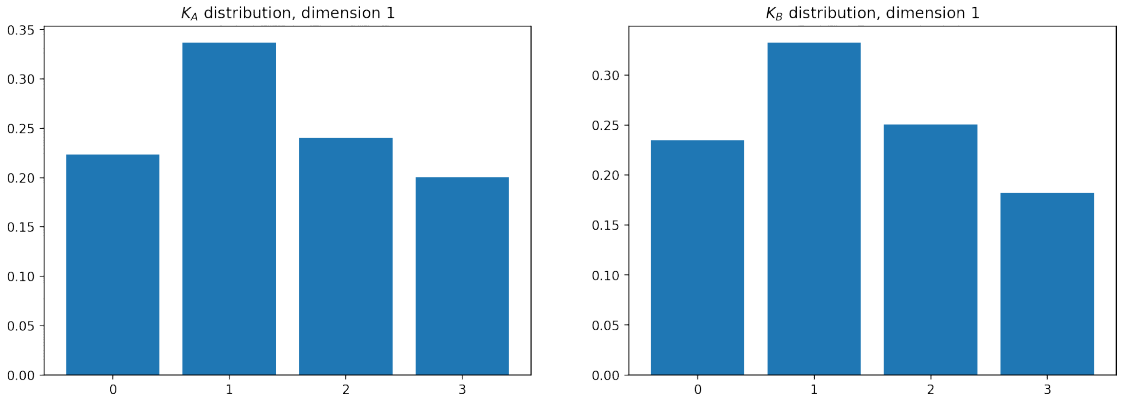


Figure 6.50: Primary key distributions for Alice (left plot) and Bob (right plot), underwater data set.

6.2.8 Comparison between NN architectures

As we have done in Section 6.1.8, we plot the SKR curves of all architectures for each value of $\mathrm{b} \in \{1, 2, 3, 4\}$ together to better understand the differences.

More particularly, Fig. 6.54 compares the SKR of the AE+UQ, AE+DSQ, and AAE+UQ architectures. For $\mathrm{b} = 1$, the best model is the AE+DSQ, while the worst one is the AE+UQ. For $\mathrm{b} = 2, 3$, the best is the AAE, while it is the worst for $\mathrm{b} = 4$, and the reason for that has been already explained in Section 6.1.7.

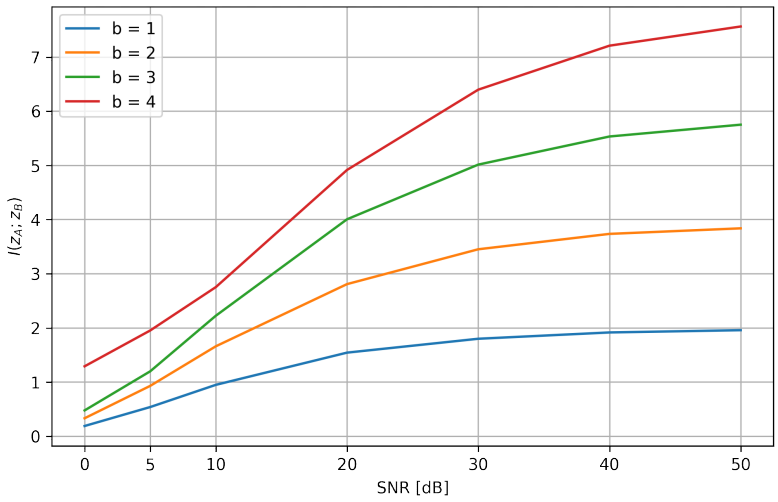


Figure 6.51: Mutual information between Alice and Bob for the AAE, underwater data set.

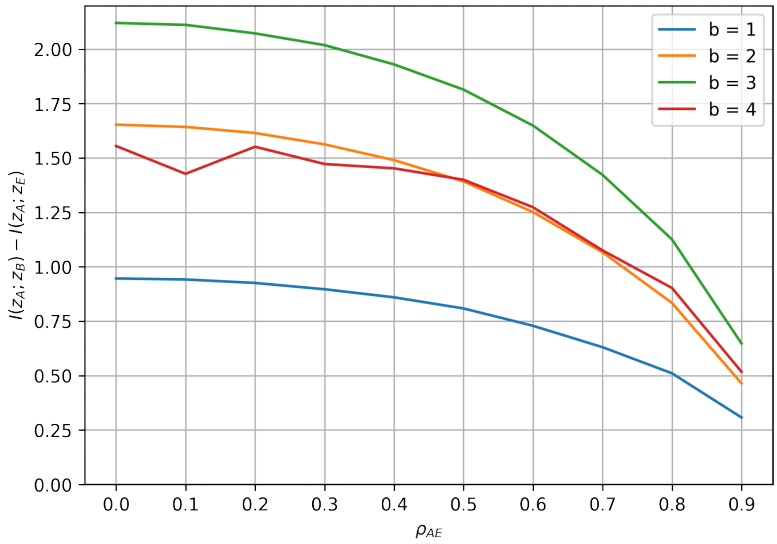


Figure 6.52: SKR between Alice and Bob for the AAE, underwater data set.

Instead, in Fig. 6.54 we have the SKR curves of the MTAE+UQ, MTAE+UQ, DAAE+UQ, DAAE+DSQ, and AAE+UQ (with eavesdropper-aware training). The best network here is the AAE, apart from the case $\mathrm{b} = 4$ which is still problematic.

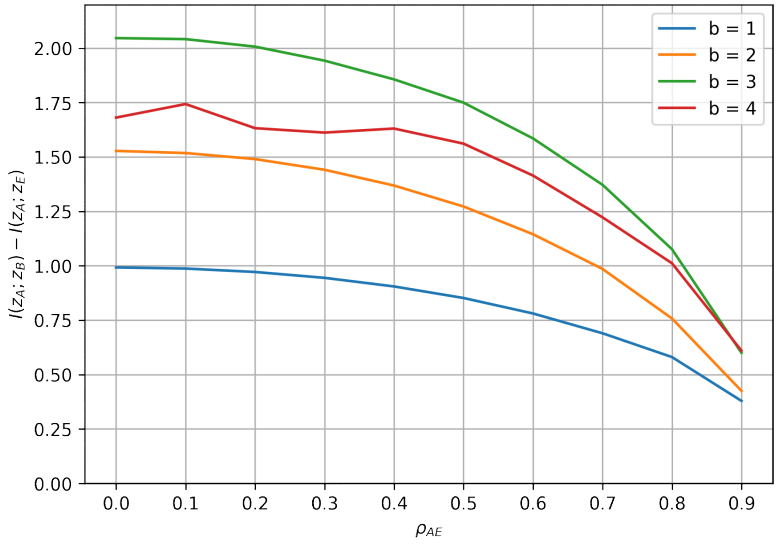
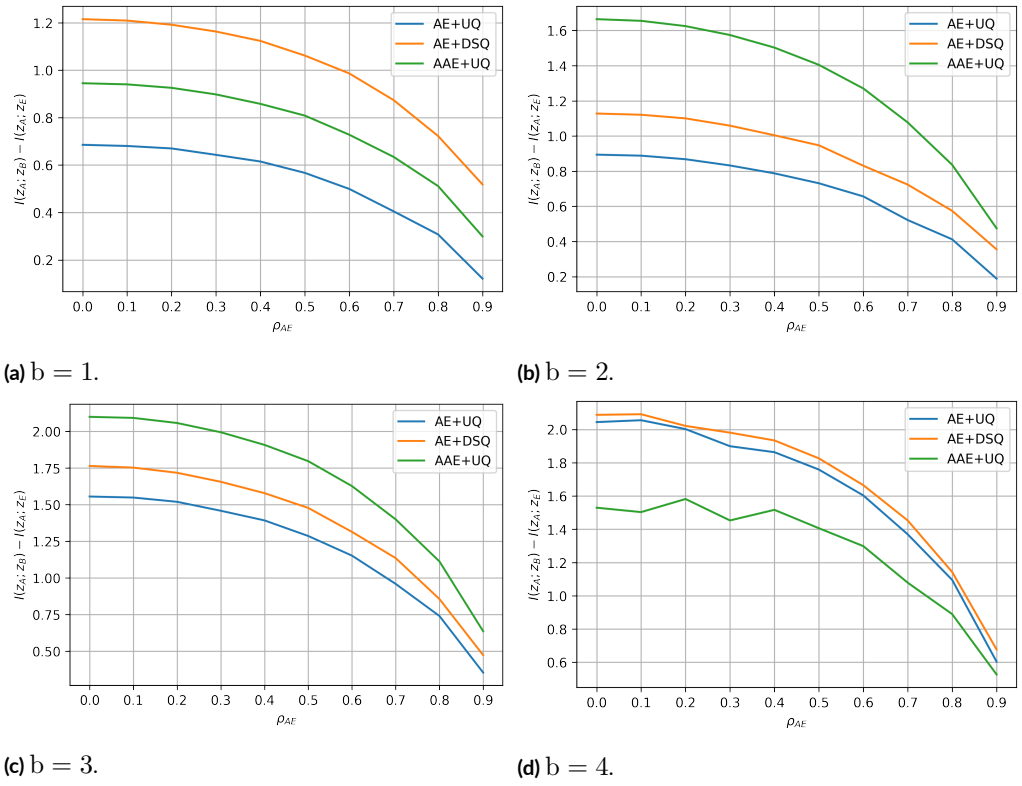


Figure 6.53: SKR between Alice and Bob for the AAE with eavesdropper-aware training, underwater data set.



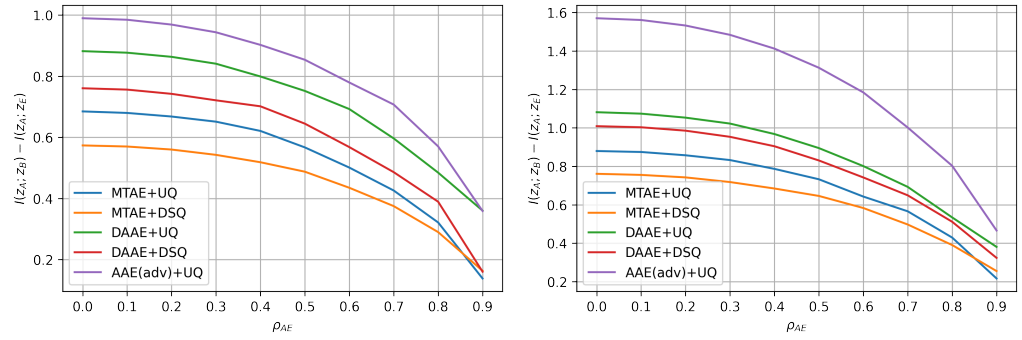
(a) $\mathrm{b} = 1.$

(b) $\mathrm{b} = 2.$

(c) $\mathrm{b} = 3.$

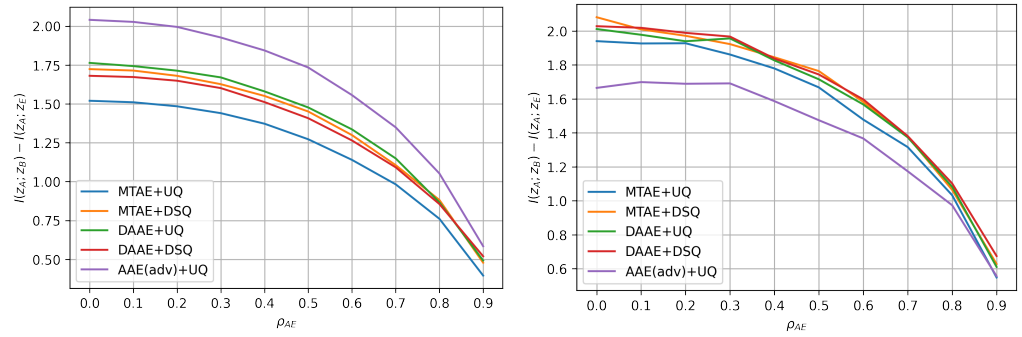
(d) $\mathrm{b} = 4.$

Figure 6.54: Comparison of SKR curves of AE+UQ, AE+DSQ, and AAE+UQ, underwater data set.



(a) $\mathrm{b} = 1$.

(b) $\mathrm{b} = 2$.



(c) $\mathrm{b} = 3$.

(d) $\mathrm{b} = 4$.

Figure 6.55: Comparison of SKR curves of MTAE+UQ, MTAE+DSQ, DAAE+UQ, DAAE+DSQ, and AAE(adv)+UQ, underwater data set.

7 Conclusion

This thesis addressed the quantization problem via machine learning and neural network approaches. For the first paradigm we have designed a joint non-uniform quantizer for Alice and Bob in such a way that in the exploitation phase the legitimate parties can independently generate an initial key based solely on the observation of their channel features. The first version of the algorithm accepted only one feature per user, which is not ideal from the key security point of view. For this first attempt we also investigated its design taking into account the operations done bye the eavesdropper (Eve). Subsequently, we have extended the joint quantizer to accept an arbitrary number of features per user basing ourselves on the SVM paradigm, i.e., each user space is divided into a given number of region with hyperplanes. As a future activity, it would be interesting to discover a way to perform the adversarial training also with this generalized version.

Regarding the neural network approach, we have proposed an autoencoder trained with a soft version of a uniform quantizer (the DSQ layer) between encoder and decoder to make the neural network distribute the latent space in such a way that, in the exploitation phase, there is little loss when applying a hard uniform quantizer to the output of the encoder to rectify the output of the DSQ layer. We compared it with the same architecture trained in the real domain and with a uniform quantizer in exploitation to produce the initial key.

However, this architecture had two downsides: it does not consider the presence of an eavesdropper, and it produces an initial key with poor randomness. To address the first problem, we have implemented two state-of-the-art architectures, the MTAE and the DAAE, and we have

seen that the eavesdropper-aware training reduces the information regarding Eve in the initial key. Furthermore, we have equipped and trained the MTAE and DAAE with the DSQ layer, and this time we saw some improvements with respect to the same architectures trained in the real domain.

To address the second issue we have proposed another architecture, the AAE, that does not use DSQ. More particularly, this architecture is trained to reshape the latent space distribution of an autoencoder and in our case we want that to be as close as possible to a uniform distribution. In this way, in the exploitation phase, we can optimally apply the hard uniform quantizer to the output of the encoder to produce the initial key. From the results obtained we saw a considerable gain in the mutual information between legitimate parties. We also trained the AAE in an adversarial manner to limit Eve's influence. In this way, the secret key rate is better than the naive autoencoder and we also have the high randomness of the key, making the AAE better than the MTAE and DAAE.

References

[1] L. Wang, H. An, H. Zhu, and W. Liu, "Mobikey: Mobility-based secret key generation in smart home," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7590–7600, 2020.

[2] J. Han, X. Zeng, X. Xue, and J. Ma, "Physical layer secret key generation based on autoencoder for weakly correlated channels," in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020, pp. 1220–1225.

[3] J. Zhou and X. Zeng, "Physical layer secret key generation for spatially correlated channels based on multi-task autoencoder," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022, pp. 144–150.

[4] ——, "Physical-layer secret key generation based on domain-adversarial training of autoencoder for spatial correlated channels," *Applied Intelligence*, Jun 2022. [Online]. Available: https://doi.org/10.1007/s10489-022-03777-w

[5] Y. Huang, S. Zhou, Z. Shi, and L. Lai, "Channel frequency response-based secret key generation in underwater acoustic systems," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 5875–5888, 2016.

[6] G. Yang, L. Dai, and Z. Wei, "Challenges, threats, security issues and new trends of underwater wireless sensor networks," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3907

[7] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," in *Proceedings of the 1st ACM International Workshop on Underwater Networks*, ser. WUWNet '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 41–47. [Online]. Available: https://doi.org/10.1145/1161039.1161049

[8] W. H. Thorp, "Analytic description of the low-frequency attenuation coefficient," *Journal of the Acoustical Society of America*, vol. 42, pp. 270–270, 1967.

[9] F. H. Fisher and V. P. Simmons, "Sound absorption in sea water," *The Journal of the Acoustical Society of America*, vol. 62, no. 3, pp. 558–564, 1977. [Online]. Available: https://doi.org/10.1121/1.381574

[10] G. Burrowes and J. Y. Khan, "Short-range underwater acoustic communication networks," in *Autonomous Underwater Vehicles*, N. A. Cruz, Ed. Rijeka: IntechOpen, 2011, ch. 8. [Online]. Available: https://doi.org/10.5772/24098

[11] C. Gussen, P. Diniz, M. Campos, W. Martins, F. Costa, and J. Gois, "A survey of underwater wireless communication technologies," *Journal of Communication and Information Systems*, vol. 31, no. 1, Oct. 2016. [Online]. Available: https://jcis.sbrt.org.br/jcis/article/view/362

[12] M. Bloch and J. Barros, *Physical-Layer Security: From Information Theory to Security Engineering*. Cambridge University Press, 2011.

[13] J. Zhang, T. Q. Duong, A. Marshall, and R. Woods, "Key generation from wireless channels: A review," *IEEE Access*, vol. 4, pp. 614–626, 2016.

[14] X. Zhang, G. Li, J. Zhang, A. Hu, Z. Hou, and B. Xiao, "Deep-learning-based physical-layer secret key generation for fdd systems," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6081–6094, 2021.

[15] A. V. Guglielmi, A. Muraro, G. Cisotto, and N. Laurenti, "Information theoretic key agreement protocol based on ecg signals," *arXiv preprint arXiv:2105.07037*, 2021.

[16] L. Bragagnolo, F. Ardizzon, N. Laurenti, P. Casari, R. Diamant, and S. Tomasin, "Authentication of underwater acoustic transmissions via machine learning techniques," in *2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, 2021, pp. 255–260.

[17] F. Ardizzon, R. Diamant, P. Casari, and S. Tomasin, "Machine learning-based distributed authentication of uwan nodes with limited shared information," *arXiv preprint arXiv:2208.09340*, 2022.

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[19] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4852–4861.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

Acknowledgments

I want to express my gratitude to Professor Stefano Tomasin to have made this research possible, but also to have been present throughout all the work an guided me with his knowledge. Special thanks also to Francesco Ardizzon, who has supported me when I did not fully understand something, and to Laura Crosara, who has helped me with the implementation of the DSQ function.

Then, I would like to thank Gloria who has been to my side during moments of discouragement and spurred me to get back up. Last but not least, I also want to thank my parents that if it were not for them, I could never have the possibility to concentrate solely on my studies and complete them in time.