Department of Information Engineering and Computer Science

Master's Degree in
Computer Science

Final Dissertation

MIMO physical layer security using multiple Reconfigurable Intelligence Surfaces

A study in vehicular environments

Supervisor
Segata Michele

Co-Supervisor
Casari Paolo

Student
Marrocco Simone

Academic year 2024/2025

Acknowledgements

...thanks to...

Contents

Abstract

This paper presents an extension of physical layer security techniques using Reconfigurable Intelligent Surfaces (RISs) in Multiple-Input Multiple-Output (MIMO) communications. Building upon previous work on secure transmissions, we generalize the mathematical framework to support multiple legitimate receivers and complex RIS configurations and combinations, including parallel and in series reflections. Our approach maintains low error rate for legitimate users while ensuring high levels of artificial noise for eavesdroppers, preserving security through Space Shift Keying (SSK) modulation. Extensive simulations analyze Bit Error Rate (BER) performance across various scenarios and demonstrate that our framework achieves robust security and relaiability. Our simulations include realistic channel modeling, incorporating Rician fading and different path loss calculations, and presents them through BER heatmaps. The proposed framework offers promising applications for emerging technologies requiring secure communication, such as vehicular networks and Internet of Things, without introducing the latency overhead of complex encryption schemes.

1 Introduction

Everyday, more and more people and objects connect, communicate and transfer data with each other. In today's world, we need both speed and security for our communications, even if both were considered two opposite ends of a spectrum. By adding error correction and encryption to our data, we can increase reliability and privacy at the expense of latency.

Modern technologies, like the Internet of Things (IOT) and the Cooperative Autonomous Driving (CAV), are becoming more and more popular and necessary in our society. But they are highly demanding advancements, needing real time communication and defence against dangerous distruptions.

Reconfigurable Intelligent Surfaces (RIS) are a new proposal that may help in this context. They are a low power, low cost solution to transform reflection from passive noise in our communications into active parameters we can fine tune to redirect, expand and propagate our communication signals into more complex scenario, for example by helping in situation without direct Line of Sight (LOS).

In this thesis, we aid the research by expanding current literature and studying how to use RIS not only for the aforementioned reason, but also to protect our communication privacy against malicious eavesdroppers. We can modulate the reflection signal to make sure only the legitimate receivers can understand the message, while making the reflected signal be undeciphrable and act as artificial noise to protect against unwanted listeners.

This is called Physical Layer Security (PLS): our objective is supporting higher levels of security, like encryption, to protect ourself against adversaries actors even when they have bigger resources than us, or reduce the complexity of it by ensuring less probability of capturing the signal in the first place.

Thank to multiple antennas communications, called Multiple Input Multiple Output, we can activate only a certain subset of it at any given time, and modulate the reflection so that the subset is catchable only by specific users, while having other eavesdroppers make random guesses.

1.1 Our contribution

The specific contributions of this work are:

- a general explanation of the current advancements in Physical Layer Security and Reconfigurable Intelligent Surfaces

- a detailed explanation of a proposed signal modulation, called Space Shift Keying (SSK), and a proposed framework for RIS-aided encryption made by promiment researcher in the field
- generalize the framework to support multiple legitimate, multiple RIS in series, and multiple signal paths in parallel, while keeping the same level of security and complexity
- carry out Bit Error Rate (BER) simulation analysis to prove the efficancy of our proposed solution
- model a realistic communication system to include channel gain matrix calculations, Rician fading and path loss
- in particular, study different configurations of path losses to give a better general vision of the applicability of our solution
- heatmap graphs to show the behavior in various practical scenarios

We will provide detailed mathematical explanation and extensive simulation code to better help the research in more future application studies about the application in real life communications between physical actors.

1.2 Notation

We will use the following notations in this work:

- Variables are written as capital italic letters X
- Vectors are written as italic letters x
- Matrixes are written as bold capital italic letters $\boldsymbol{X}$
- $\mathbb{C}$ defines the Complex set, C^X a complex vector of lenght X, and C^{XxY} a complex matrix of dimension X rows and Y columns
- given $x \in \mathbb{C}^Y$, we define $\boldsymbol{X} = diag\{p\} \in \mathbb{C}^{YxY}$ a matrix with all zero, except in the diagonal where position y, y is equal to x_y
- given $\boldsymbol{X} \in \mathbb{C}^{YxY}$, we define $x = diag(\boldsymbol{X}) \in C^Y$ the vector of the elements in the diagonal of $\boldsymbol{X}$
- given $\boldsymbol{X} \in \mathbb{C}^{YxY}$, we define $\boldsymbol{X}_{diag} \in \mathbb{C}^{YxY}$ the matrix with all zero, except in the diagonal where position y, y is equal to $\boldsymbol{X}_{y,y}$
- given $\boldsymbol{X} \in \mathbb{C}^{YxZ}$, we define $[\boldsymbol{X}]_{:,1:Y} \in \mathbb{C}^{YxY}$ the first Y columns of $\boldsymbol{X}$
- $\odot$ is the Hadamard product
- An hermitian transpose of $\boldsymbol{V}$ ($\boldsymbol{V}^H$), means we fist transpose the matrix ($\boldsymbol{V} \rightarrow \boldsymbol{V}^T$), then take the conjugate of every element (so invert the sign of the immaginary part).
- The Frobenius norm of a matrix $\boldsymbol{X}$, denoted as $\|\boldsymbol{X}\|$, is defined as $\|\boldsymbol{X}\| = \sqrt{\sum_i \sum_j |x_{ij}|^2}$

2 Related works

2.1 Physical Layer Security

The essential premise of physical layer security is to enable the exchange of confidential messages over a wireless medium in the presence of unauthorized eavesdroppers, without relying on higher-layer encryption. [22]

Physical layer security is much lighter than complex secret key-based cryptographic technique [23]. It ensures high reliability and security at less computational cost. Being much quicker than the higher layer counterparts, it ensures low latency in higher bands, like 5G or 6G. Ultra-reliable low latency communication (URLLC) may help critical infrastructure in deliverying high privacy and data speed capabilities. The ability to combine both lower and higher layer security will also enhance the general capabilities of everyday applications.

We have two type of threats we need to protect against. Active attacks, like jamming a frequency, distrupt and block the flow of information; while passive attacks, like eavesdropping, are more subtle and we need to make our signals undeciphrable with encryption or noise [26]. In particular, passive hearing could expose sensitive data, and even encryption may still leak location, traffic load and other sensitive informations.

Modern communication need multiple requirements in order to garantee the efficacy and security of communications:

- legitimate users should be authenticated
- access control must be implemented to ensure confidentiality of the messages
- integrity of the communication, to ensure the message is correctly delivered
- aviability of the channel link, to ensure jamming attacks do not influence negatively the flow of informations
- defences against eavesdroppers

There are different methods we can use to mask our communications: we can fingerprint the legitimate users, as explained in the paper [28], use directional antennas to reduce the area where is possible to capture the signal [9], or add artificial noise schemes to distrupt unwanted hearers [8].

Different error correction strategies can also mitigate the effect of active attacks. We can use some of the passed bit to ensure the data received is correct, and sometimes even fix the errors. We can also use *Spread Spectrum Coding* to rapidly change the frequence used to deliver the message, ensuring difficoulties in disrupting all of the aviable ones.

Modeling the threats of adversaries can be quite challenging [31]. Many research papers include assumptions that we need to be careful about. For example, eavesdroppers may not just be passive listeners, but may actually collect data to later transform into active attackers.

The adversaries may also have better resources, both in computer power and signal reception, and it is difficult to model all possible threats we may face. We may have multiple stations collaborating together in deciphering the signal, or even be backed up by national agencies. Some work has already being done adressing these issues: in [10], the authors study an universal coding scheme to protect against eavesdroppers changing constantly channels. In [15] a statistical model is created to calculate the probability of achieving secrecy from eavesdroppers in unknown locations

Achieving perfect communication secrecy is not really possible for all cases, given that we need the secret key to be at least as big as the secret message [25], but there are some practical strategies we can implement.

In particular to eavesdropping, there is a huge opportunity for improvements. While distruptions have been studied for long, especially in military communications, message encryption is usually

delegated to the higher levels [22]. However, the physical layer can assist by hiding or masking the signal, making it harder for the eavesdropper to capture it. Given the advances in quantum computing and encryption breaking algorithms [27], it is important to be protected at all layers.

2.2 Reconfigurable Intelligent Surface

Reconfigurable Intelligent Surfaces (RISs) are a new technology that can help in improving the security and reach of wireless communications. They are made of a large number of passive elements that can reflect the signals in a way that can be controlled and optimized.

With RISs, it is possible to control the propagation and reflection of radio waves, making it possible to transform the environment, in which the waves need to travel, from an uncontrollable phenomena to a programmable variable that is possible to (partially) control and optimize.

RISs can help in particular in two scenarios. In the first one, two nodes which are not in the line of sight (LOS) can communicate with the help of the RIS; in the second one, being in the LOS means an inability to take advantage of delayed reflections (especially for new technologies like 5G and 6G), which can be used to improve the signal quality and robustness, but we can create them with RISs [6].

The main advantages of RISs are the low cost, the low power consumption and the easy deployment, which makes them a good candidate for the future of wireless communications. They do not require a dedicated energy source, they do not suffer from noise amplification, they can work with any frequency and can be easly put in any surface like walls or ceilings [2].

A specific controller is used to modify dinamically the reflecting elements of the RIS, giving huge margins for custom configurations in complex scenarios and new communications frameworks.

Numerous applications are being studied to this day. For example, in [14] the authors study the modeling of path loss for a reflected signal; they then recreate a RIS using an arduino to validate the mathematical calculations.

RIS can also be included in more general smart cities projects [17]: in a more interconnected world, they can be used for everyday streets and personal homes; huge, unused building surfaces may aid the general population get better connectivity; smart factories and the industry 4.0 could be assisted in dense IoT devices places, by countering the negative effects of metals for signal strenght.

Active RIS can also be deployed [19], which can amplify the received signal before reflecting it, ensuring an even higher power output and reach for a similar design. Beamforming can also be used to redirect the signal towards a specific direction, increasing the signal strenght for the receiver in that specific area.

2.3 Using RISs for Physical Layer Security

RISs can be used to greatly increase not only the network performance but also its security [16]. By using RISs, we can make the signal quality better, reduce the signal degradation and make the signal more difficult to intercept by eavesdroppers.

For example, the reflection can be used as multiplicative randomness to make the transmission not understable from eavesdroppers, while having a decoding for the legitimate user linear [20].

Another paper [36] studied how to use a novel RIS based channel randomization technique to improve the secrecy rate, and another one [3] shows an iterative efficient algorithm to maximize the minimum secrecy rate by optimizing the reflecting coefficients of the RIS.

RISs can also be used to protect against jamming attacks: for example, in [30] it is used an aerial RIS to mitigate the effects of the disturbance and increase the transmission power and reliability.

2.4 RISs and Physical Layer Security for Vehicular Networks

Cooperative autonomous driving can bring many benefits, like reducing traffic congestion, improving road safety and reducing the environmental impact of the vehicles. Cars and other vehicles can communicate with each other and with the infrastructure to share information and coordinate their movements. However, it also brings new security concerns, especially in the wireless communications.

It is clear that it is necessary to have a secure and fast way to communicate, and 6G network technologies plus RISs can help in this regard. By reflecting the signals, we can overcome the limitations

of LOS and improve the signal quality by reducing signal degradation [5].

The sector is just starting to be studied, but there are already some promising result. Network simulators made specific, like CoopeRIS, allow to study and progress this field [24].

Vehicular networks need low latency and high security. Active attack may jeopardize drivers' and people's safety, while also slowing down information exchange rate. Being moving agents, it is more difficult to correctly model this type of network, but also way more necessary: complex upper layer encryption may slow down data processing enough to render it useless [1].

Passive attackers may instead use vehicles' geolocation and traffic data for malicious activity. A way to detect and filter out intruders is discussed in [18].

Recent studies shows how RISs can be used to protect the vehicular network against illegitimate users. In [21] the authors study how RISs can improve the average secrecy capacity and secrecy outrage probability.

2.5 Future Directions

Network intensive technologies like IOT and CAV are gaining traction fast, thanks to the many benefits they bring to society and the newest technologies that now allow this incredibly huge traffic load.

The security of these networks is still a big concern, and it is necessary to study and implement new technologies to protect them. RISs are a promising technology that can help in this regard. Being cost effective, fairly passive and easy to deploy, they can assist in overcoming the problems of 6G like signal fading and out of LOS communication [2].

However, while we have some initial literature in both physical layer security using RISs, and using RISs to improve vehicular network performances, not much has been made in studying all three of these aspects [21].

Practical solutions could be studied and simulated starting from the resources presented here. RISs can be used both to mask the network signal or to make it noisier for unwanted listeners located in different places.

For example, starting from [9], it could be studied how cooperating vehicles could calculate together with a RIS how to add noise to other locations while moving in space, and so needing constant modifications in the calculations themselves.

Modern cars have as much computing power as a modern personal computer: for example, Tesla cars have an integrated GPU to utilize the autonomus driving feature [29], which could be used for highly efficient matrix calculations [4].

In conclusion, the future of vehicular networks is bright, but it is necessary to study and implement new technologies to protect them. RISs are a promising technology that can help in this regard, and it is necessary to study how to use them to improve the security of vehicular networks.

3 Hidden comunication by targeted reflections

3.1 RIS Diagonalization

We will start from the paper *Reconfigurable Intelligent Surface: Reflection Design Against Passive Eavesdropping* [20], explaining how to hide communication between two actors from eavesdroppers using Reconfigurable Intelligent Surfaces, then expanding it to multiple receiving users at the same time.

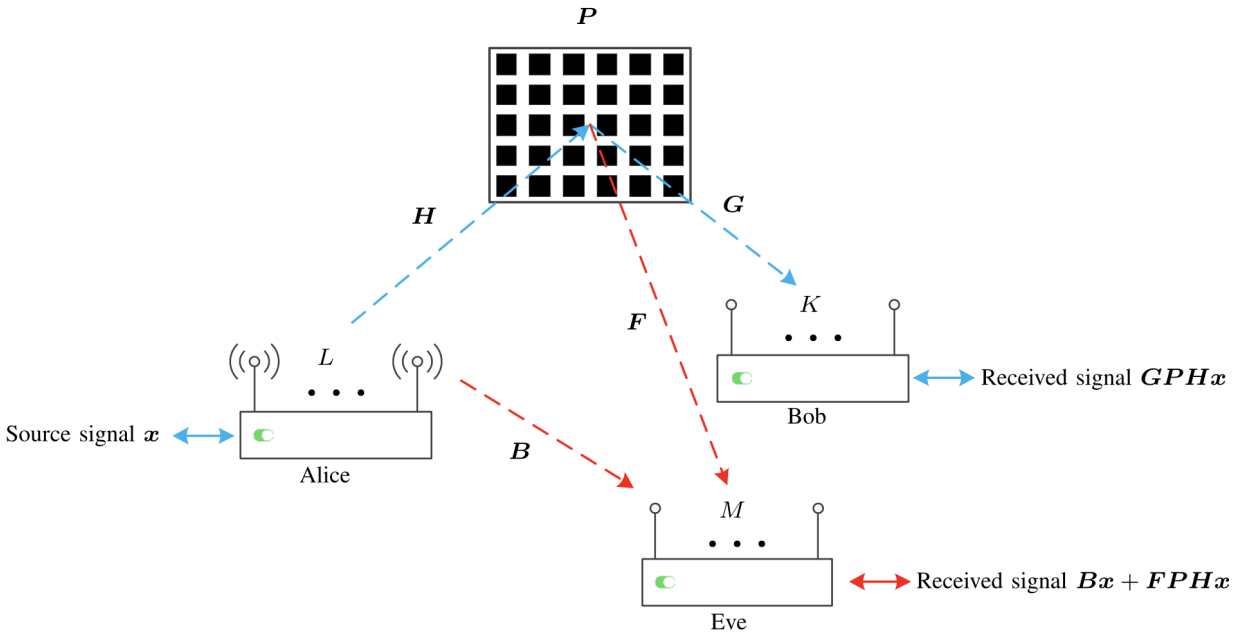

Figure 3.1: Setup

In [20], the authors studied how to use RISs to allow communications between two users without LOS, while making the signal undeciphrable for eavesdroppers. We call L the transmitter's antennas, K the receiver's antennas, M the eavesdropper's antenna, and N the RIS reflecting elements. We assume $L \geq K \geq 2$.

We define $\boldsymbol{H} \in \mathbb{C}^{NxL}$ the channel response [1] from the transmitter to the RIS, $\boldsymbol{G} \in \mathbb{C}^{KxN}$ the channel response from the RIS to the receiver, $\boldsymbol{P} = diag\{p\} \in \mathbb{C}^{NxN}$ a diagonal matrix in which the nth diagonal element represents the reflection coefficient of the nth unit at the RIS.

The objective is making the receiver's final signal $\boldsymbol{GPH}$ a diagonal matrix, while making every possible eavesdropper's final signal a full matrix.

We will leave for later the technical details of why this would achieve secrecy for the legitimate users or how the actors communicate with each others, and will just focus on the mathematics behind the calculation. It is possible to read more in the paper *Space shift keying modulation for MIMO channels* [13], which we will summarize in a later chapter.

[1] A channel response for a MIMO communication is a matrix made of complex number, where the position i, j indicates the signal received from antenna j to antenna i

Our contribution to the field will be to generalize these calculations to J receving users and M RISs used in parallel and in sequence.

Formally, the condition we want to satisfy is:

$$||[\boldsymbol{GPH}]_{:,1:K} - [[\boldsymbol{GPH}]_{:,1:K}]_{diag}||^2 = 0 \tag{3.1}$$

Where $[GPH]_{:,1:K} \in \mathbb{C}^{KxK}$ denotes the first K columns of the matrix $\boldsymbol{GPH} \in \mathbb{C}^{KxL}$.

Given

$$\boldsymbol{W} = \sum_{i,j=1,i\neq j}^{K} (g_{j,:} \odot h_i^T)^H (g_{j,:} \odot h_i^T) \tag{3.2}$$

$$\text{rank}(\boldsymbol{W}) = K(K-1) \tag{3.3}$$

$$\text{rank}(\boldsymbol{W}) - \text{null}(\boldsymbol{W}) = N \tag{3.4}$$

Where null refers to the dimension of the null space.

$$\text{null}(\boldsymbol{W}) = N - (K^2 - K) \tag{3.5}$$

The formula (3.1) can be rewritten as

$$Wp = 0 \tag{3.6}$$

and the solutions of p can be found in the null space of $\boldsymbol{W}$. Using singular value decomposition (SVD), we can decompose

$$\boldsymbol{W} = \boldsymbol{R\Sigma V}^H \tag{3.7}$$

With SVD, we have $\boldsymbol{\Sigma} = diag(\sigma) \in C^{NxN}$ a diagonal matrix. the first $\text{rank}(\boldsymbol{W}) = K^2 - K$ elements of σ are non-zero, while the last $\text{null}(\boldsymbol{W}) = N - (K^2 - K)$ elements are zero [7].

Given a more generic $\boldsymbol{A} \in \mathbb{C}^{mxn} = \boldsymbol{R'\Sigma' V'}^H$, we have the column vectors of R' being an orthonormal span of C^m, and the row vectors of V' being an orthonormal span of C^n.

Suppose $\boldsymbol{A}$ is an Hermitian matrix (meaning $\boldsymbol{A} = \boldsymbol{A}^H$). This will be useful later, as $\boldsymbol{W}$ is also an Hermitian matrix by construction. Let's call k the null space dimension of $\boldsymbol{A}$, and ,by the property above, the null space dimension of $\boldsymbol{A}^H$ too.

The last k columns of $\boldsymbol{R'}$ are a span of the null space

$$N(\boldsymbol{A}^H) = [r_{m-k}, ..., r_m] \in \mathbb{C}^{mxk} \tag{3.8}$$

while the last k rows of V'^H are a span of the null space

$$N(\boldsymbol{A}) = \begin{bmatrix} v'^H_{n-k} \\ ... \\ v'^H_n \end{bmatrix} \in \mathbb{C}^{kxn} \tag{3.9}$$

Being $\boldsymbol{A}$ an Hermitian matrix, the two null spaces are both solutions to $\boldsymbol{A}x = 0$.

Consider now $\boldsymbol{W} \in C^{NxN}$. The paper in question uses equation (3.8) to find the solutions, since W is hermitian and square. Taken $\boldsymbol{U} \in \mathbb{C}^{Nx(N-(K^2-K))}$ the last $N - (K^2 - K)$ columns of the left singular matrix $\boldsymbol{R}$. $\boldsymbol{U} \in \boldsymbol{N}(\boldsymbol{W})$ for the explanation above. We then have

$$\boldsymbol{WU} = 0 \tag{3.10}$$

$$p = \boldsymbol{U}q \tag{3.11}$$

$$\boldsymbol{WU}q = 0 \tag{3.12}$$

being true, and $q \in \mathbb{C}^{N-(K^2-K)}$ can be a random vector.

3.2 Space Shift Keying Modulation

How can the actors communicate, if the result is a diagonal matrix with random value?

We will use a technique called *Space Shift Keying* (SSK) Modulation [13], where *antenna indices are used as the only means to relay information.* Given K the number of antenna of the actors in the system, we can send $log_2(K)$ bits by mapping each combination of bits to a specific antenna. [2]

TABLE I
EXAMPLE OF THE SSK MAPPER RULE.

$\mathbf{b} = [b_1 \quad b_2]$	symbol	antenna index j	$\mathbf{x} = [x_1 \quad \cdots \quad x_4]^T$
$[0 \quad 0]$	0	1	$[1 \quad 0 \quad 0 \quad 0]^T$
$[0 \quad 1]$	1	2	$[0 \quad 1 \quad 0 \quad 0]^T$
$[1 \quad 0]$	2	3	$[0 \quad 0 \quad 1 \quad 0]^T$
$[1 \quad 1]$	3	4	$[0 \quad 0 \quad 0 \quad 1]^T$

Figure 3.2: SSK conversion table, from [13]

3.2.1 Direct Detection

Given a channel gain matrix $\boldsymbol{B} \in \mathbb{C}^{KxK}$ and the input vector $x \in \mathbb{C}^K$ with only one element equal to 1, the signal received is

$$y = \boldsymbol{B}x + \sigma^2 \tag{3.13}$$

To understand the antenna index which sent the message, we need to find the column b_j which is most similar to y.

$$j = \arg\max_j \; p_y(y|x_j, \boldsymbol{B}) = \arg\min_j \; ||y - b_j||^2 \tag{3.14}$$

3.2.2 Diagonalized Reflection Detection

Following [20], for a reflected signal we have

$$y = \boldsymbol{GPH}x + \sigma^2 \tag{3.15}$$

Given that $\boldsymbol{GPH}$ is a diagonal matrix and x has only one element equal to 1, the resulting vector $\boldsymbol{GPH}x$ will still be a vector with only one element non zero. Adding noise, to find the antenna index we search for the biggest value in the vector.

$$j = \arg\max_j \; y_j \tag{3.16}$$

3.3 Cascaded Channel Estimation

To understand how the actors (and in particular the RISs controller) estimate the channel gain between them, we redirect to the paper *Cascaded Channel Estimation for Large Intelligent Metasurface Assisted Massive MIMO* [11]. While we will not summarize the content here, we will still give a general idea of how to use the algorithm in the paper to estimate $\boldsymbol{G}$ and $\boldsymbol{H}$.

[2] This may seem rather unoptimized, as we use only one antenna instead of combinations of them. To see a more general approach, the authors also wrote the paper [12], where they discuss a more general approach using multiple active antennas at the same time. The general approach will also work with our proposed solution.

- The transmitter comunicates to the RIS controller a setup message x' that it will send to the receiver;
- The RIS will set a random $\boldsymbol{P}'$;
- The receiver gets a signal y' (which will mean nothing), and sends it back to the RIS controller;
- Based on $x', y', \boldsymbol{P}'$ the RIS controller estimates $\boldsymbol{G}, \boldsymbol{H}$ and correctly setup $\boldsymbol{P}$;
- The transmitter sends x, and the receiver gets y which can correctly convert back;
- If transmitter and receiver are moving, the procedure will start all over. Otherwise, $\boldsymbol{G}$ and $\boldsymbol{H}$ remain the same, and the RIS controller can just create a new $\boldsymbol{P}$ for the next messages.

4 Expanding to multiple users

In real life scenarios, we deal with more than two communicating actors. We want to expand the findings of this paper by having it support multiple RISs in series and multiple receivers from the same transmitter. Once we have those, we can generalize it to also have receivers getting signals from multiple indipendent reflections of RISs.

We will first, however, make some semplifications about the actors by having $L = K$ for all of them[1]. We will still consider one transmitter, with $J \geq 1$ receivers.

4.1 Reflecting to multiple users

We consider the case where the transmitter wants to send the signal to J receivers without LOS. The condition we want to satisfy is

$$\forall j \in [1...J] \rightarrow ||\boldsymbol{G}_j \boldsymbol{P} \boldsymbol{H} - [\boldsymbol{G}_j \boldsymbol{P} \boldsymbol{H}]_{diag}||^2 = 0 \tag{4.1}$$

$$\forall j \in [1...J] \rightarrow \boldsymbol{W}_j p = 0 \tag{4.2}$$

$$\begin{bmatrix} \boldsymbol{W}_1 \\ \boldsymbol{W}_2 \\ ... \\ \boldsymbol{W}_j \end{bmatrix} p = 0 \tag{4.3}$$

$$\begin{bmatrix} \boldsymbol{W}_1 \\ \boldsymbol{W}_2 \\ ... \\ \boldsymbol{W}_j \end{bmatrix} = \boldsymbol{W} \in \mathbb{C}^{JNxN}, \boldsymbol{W} = \boldsymbol{R} \boldsymbol{\Sigma} \boldsymbol{V}^H \tag{4.4}$$

The problem we have now is that $\boldsymbol{W}$ is not a square matrix anymore, so we cannot use the last $N - (K^2 - K)$ columns of $\boldsymbol{R}$ to calculate the null space and p with its linear combination, by using formula (3.8). The null space would have dimension $N(\boldsymbol{W}) \in \mathbb{C}^{JNx(N-(K^2-K))}$, but that would require a $p \in C^{JN}$ instead of $p \in C^N$.

We can, however, use the the last $N - (K^2 - K)$ rows of $\boldsymbol{V}^H$, then apply again the hermitian transposition to get our desired solution. Remember that $N(\boldsymbol{W})$ can also be calculated using the left singular matrix, by using formula (3.9). $\boldsymbol{W}$ is not a square matrix, so $\boldsymbol{W} \neq \boldsymbol{W}^H$,

$$N(\boldsymbol{W}) = \begin{bmatrix} v^H_{N-J(K^2-K)} \\ ... \\ v^H_N \end{bmatrix}^H \tag{4.5}$$

Take $\boldsymbol{U}_1 \in \mathbb{C}^{N-J(K^2-K)xN}$ the last $N - (K^2 - K)$ rows of $\boldsymbol{V}^H$, and

$$\boldsymbol{U} = \boldsymbol{U}_1^H \in \mathbb{C}^{NxN-J(K^2-K)} \tag{4.6}$$

We now can apply the same method as before

[1] We want the actors to be able to communicate with each other. Since the transmitter needs to have an equal or greater number of antenna than the receiver, but the roles may later switch, it follows that the number of antennas must be equal for the calculation. Using more antennas can still be done, by not considering the values coming from them (like the original paper did as well).

$$p = \boldsymbol{U}q \tag{4.7}$$

$$\boldsymbol{W}\boldsymbol{U} = 0 \tag{4.8}$$

$$\boldsymbol{W}\boldsymbol{U}q = 0 \tag{4.9}$$

4.2 RISs in parallel

Given the previous property, it follows that we can use M indipendent RIS, each one reflecting the signal to J multiple receivers, and without LOS from each other. For the receiver $j \in [1, J]$, we have

$$\sum_{m=1}^{M} \boldsymbol{G}_j \boldsymbol{P}_m \boldsymbol{H}_m x = (\sum_{m=1}^{M} \boldsymbol{G}_j \boldsymbol{P}_m \boldsymbol{H}_m)x \tag{4.10}$$

The sum of diagonal matrixes is still a diagonal matrix, so the property still holds. Remember that we only care about the indexes of the active antennas and not their values, so there is no problem in adding them together.

4.3 RISs in series

We consider the case where the signal is bounced between M RISs in this way:

$$\text{Transmitter} \rightarrow \text{RIS 1} \rightarrow ... \rightarrow \text{RIS M} \rightarrow \text{Receiver} \tag{4.11}$$

We call $\boldsymbol{C}_i \in \mathbb{C}^{NxN}$ the channel gain between $\boldsymbol{P}_i$ and $\boldsymbol{P}_{i+1}$. We need to solve

$$||\boldsymbol{G}\boldsymbol{P}_1\boldsymbol{C}_1...\boldsymbol{P}_M\boldsymbol{H} - [\boldsymbol{G}\boldsymbol{P}_1\boldsymbol{C}_1...\boldsymbol{P}_M\boldsymbol{H}]_{diag}||^2 = 0 \tag{4.12}$$

We can generate $p_1, ..., p_{M-1}$ as random reflections, and calculate the last one based on the previous. An advantage we get is that eavesdroppers listening from a middle RIS will not be able to decipher the signal either.

Given $r_i \in [0, 1]$ the absorption coefficient, and $\theta_i \in [0, 2\pi]$ the phase shift, we can choose them randomly for all RIS p_m vectors, but the last one.

$$\forall m \in [1, M-1] : p_m[i] = \eta * r_i * e^{j\theta_i} \tag{4.13}$$

Given now

$$\boldsymbol{G}' = \boldsymbol{G}\boldsymbol{P}_1\boldsymbol{C}_1...\boldsymbol{P}_{M-1}\boldsymbol{C}_{M-1} \in \mathbb{C}^{KxN} \tag{4.14}$$

We can consider now the problem of solving

$$||\boldsymbol{G}'\boldsymbol{P}_M\boldsymbol{H} - [\boldsymbol{G}'\boldsymbol{P}_M\boldsymbol{H}]_{diag}||^2 = 0 \tag{4.15}$$

Which can be solved as before. [2] [3]

If we have multiple $\boldsymbol{G}_j$, it will be enough to calculate all the $\boldsymbol{G}'_j$ and proceed as before, allowing us to combine these properties in more complicated scenarios.

[2] It is also possible to set up randomly the last $M-1$ RIS and calculate the first one using $\boldsymbol{G}$ and $\boldsymbol{H}' = \boldsymbol{C}_1\boldsymbol{P}_2\boldsymbol{C}_2...\boldsymbol{P}_M$. The properties still holds.

[3] Estimate the channel gains $\boldsymbol{G}$ and $\boldsymbol{H}$, based on [11], could be more difficult, given that we do not have full controll on $\boldsymbol{P} = \boldsymbol{P}_1\boldsymbol{C}_1...\boldsymbol{P}_M$ anymore. We can however estimate directly G' by keeping the same random $\boldsymbol{P}_1, ..., \boldsymbol{P}_{M-1}$ in both the acknowlegdment round and the message trasmission round, and just modify $\boldsymbol{P}_M$ after estimating $\boldsymbol{G}'$ and $\boldsymbol{H}$ to correctly deliver the message.

4.4 Complex reflections

The receiver could also get the signal from all the RIS in series, if in the right position.

For example, let's say it receives the signals $\boldsymbol{GP}_1\boldsymbol{H}_1x$ and $GP_1C_1P_2H_2x$. To solve this system, instead of setting $\boldsymbol{P}_1$ randomly, we would need first to solve it using $\boldsymbol{G}$ and $\boldsymbol{H}_1$, then solve $\boldsymbol{P}_2$ using $\boldsymbol{G}' = \boldsymbol{G}\boldsymbol{P}_1\boldsymbol{C}_1$ and $\boldsymbol{H}_2$. The sum of the two signals would still be readable for the receiver correctly.

While the calculations of $\boldsymbol{P}_1$ depends on $\boldsymbol{G}$ and $\boldsymbol{H}_1$, the signal would still be random and undecipherable for an eavesdropper receiving it.

5 Simulation Results

We can now test more complex setups by doing Bit Error Rate (BER) analysis.

5.1 BER stocastic simulation

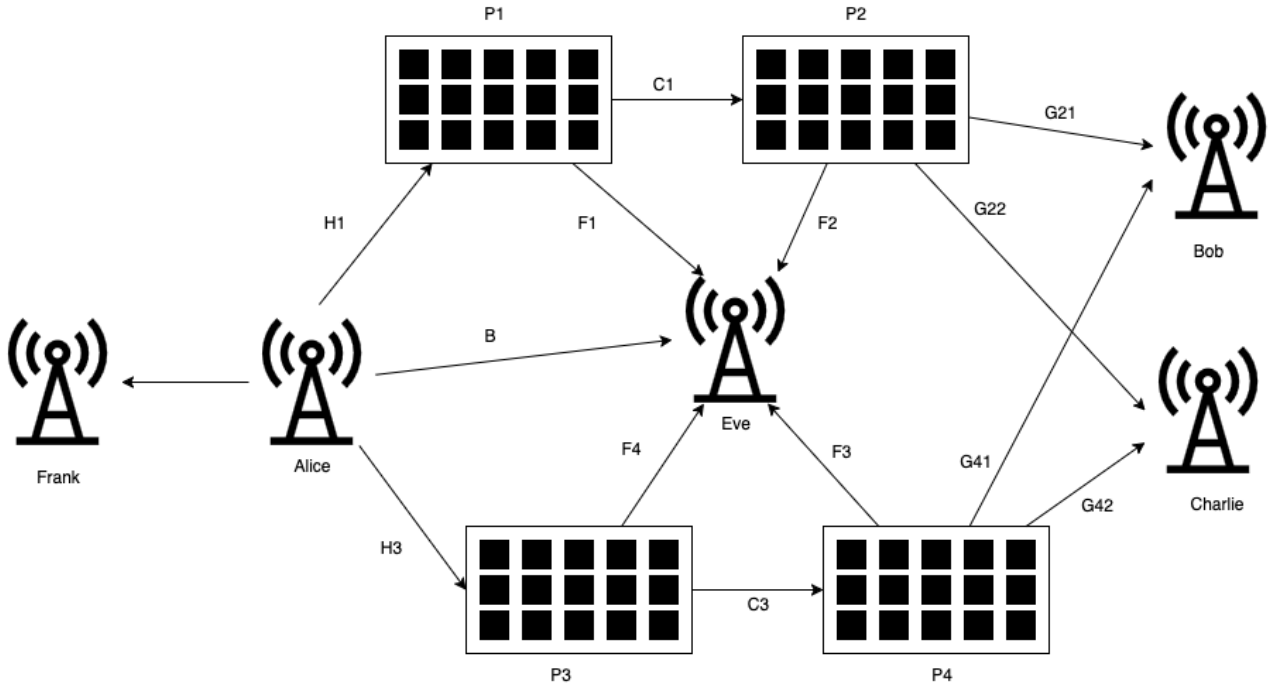

Figure 5.1: Complex Setup

For example, here we have a single transmitter *Alice* and multiple receivers. *Frank* is a direct receiver in line of sight. *Bob* and *Charlie* receive the signal from two double RIS reflection. *Eve* receives both the direct signal and the reflecting signal from all RISs. [1]

More in general, we would have M consecutive RIS (in series) that reflect a signal, J legitimate receivers and Q different paths of RIS (in parallel) to send the signal at the same time. [2]

We will show simulation results for different combinations of (M, J), both with a single and double path. In all scenarios, $K = 2, N = 16, \eta = 0.9$ will be the number of antennas for all actors, the number of reflecting surfaces and the reflection coefficients. We take these parameters to compare the results to the original paper [20].

The direct link and the eavesdropper will try to understand the message by following the equation (3.14), while the receivers will try to understand it by following the equation (3.16).

[1] It should be noted that if *Eve* is in the same position as *Frank* and receives just the direct signal, our particular framework would not give us physical layer security, and higher layer security would be needed. If instead *Eve* has not line of sight, the message would be completely unreadable from the start, since it would receive random matrixes.

[2] The paths could have a different number of RIS (for example, a path of three and another of two). The results would still hold.

5.1.1 Single RIS reflection (M=1)

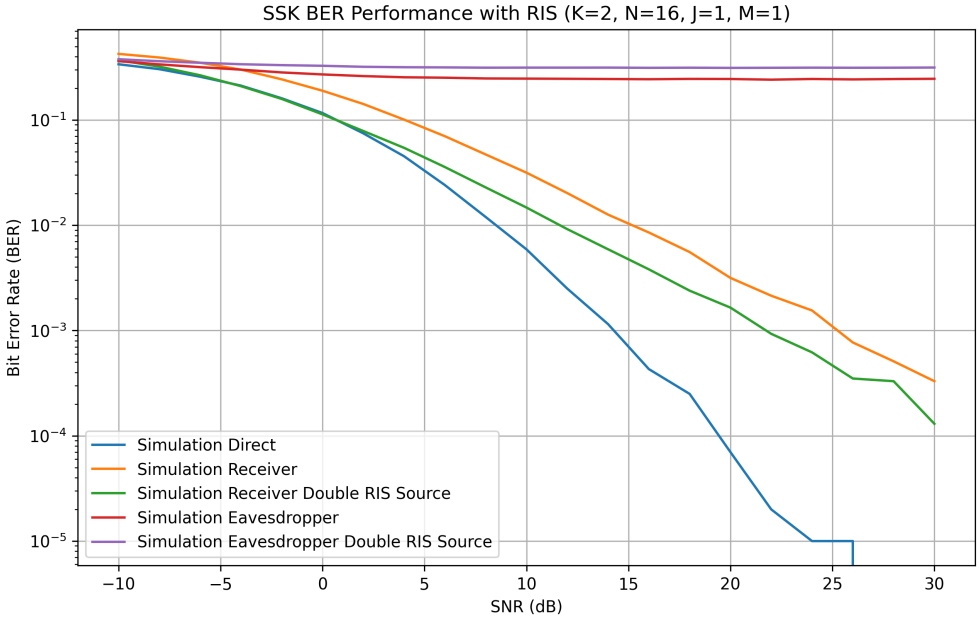

Figure 5.2: SSK BER Performance with RIS (K=2, N=16, J=1, M=1)

We can see in ($M = 1, J = 1$) the results match with [20], for both *Simulation Receiver* and *Simulation Eavesdropper*. *Simulation Direct* is the strongest possible path, mainly because of the reflection loss due to η. Combining two different RIS in parallel (*Double RIS Source*) gives better signal to the receiver, while disturbing more the signal to the eavesdropper.

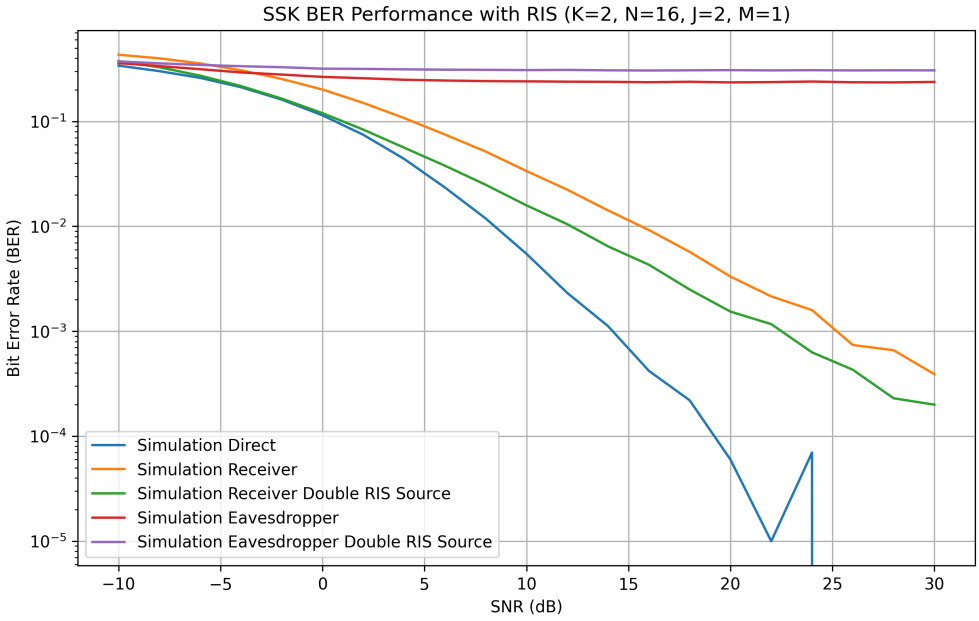

Figure 5.3: SSK BER Performance with RIS (K=2, N=16, J=2, M=1)

Increasing the number of receivers does not influence the result of our framework: the receivers still get a good signal depending on the SNR, while the eavesdropper is not getting an advantage in understanding the message.

5.1.2 Double RIS reflection (M=2)

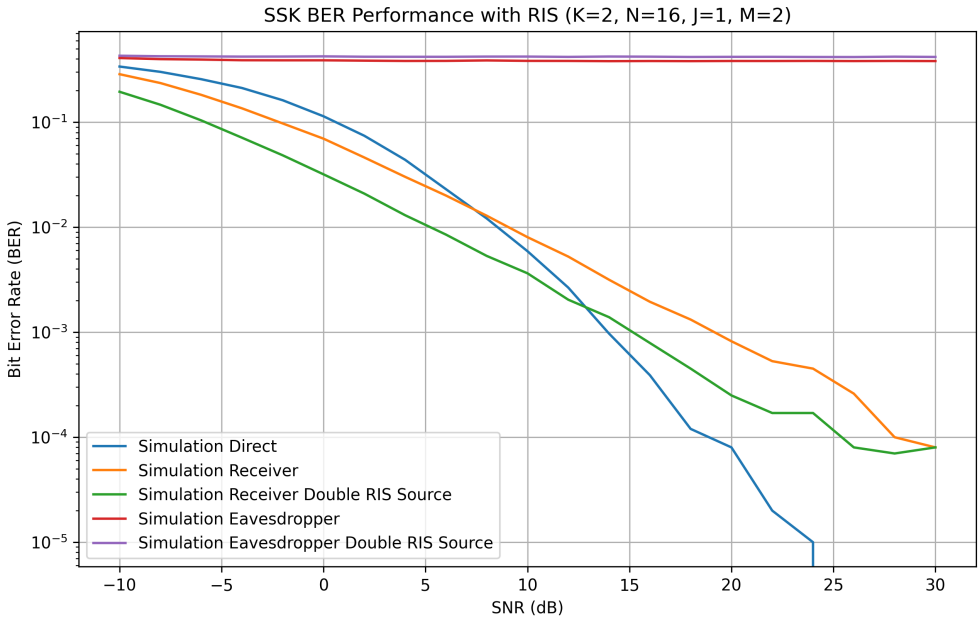

Figure 5.4: SSK BER Performance with RIS (K=2, N=16, J=1, M=2)

With multiple RIS in series, the eavesdropper get a worse signal because of the double interference of the 2 RIS.

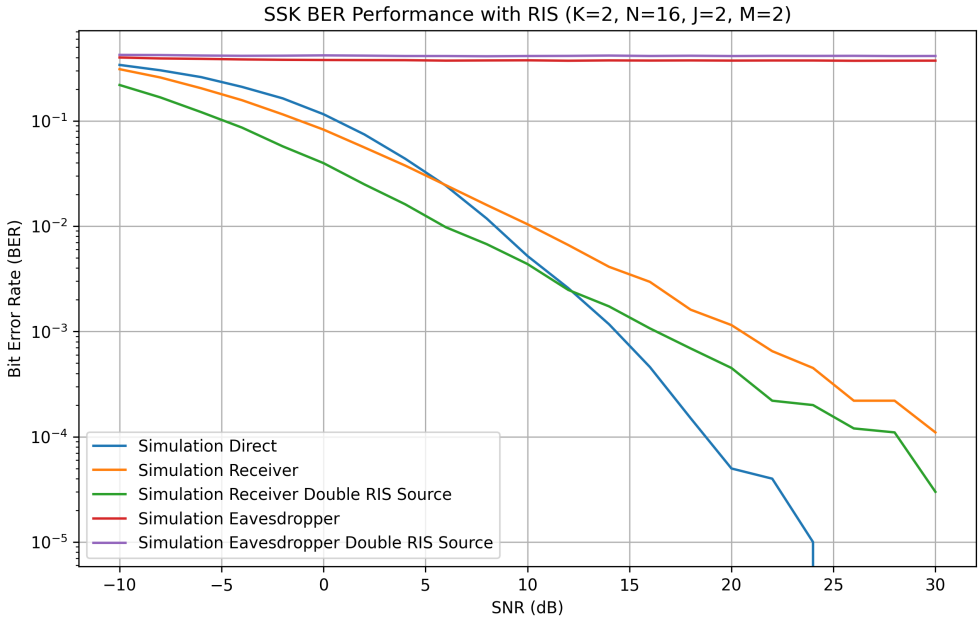

Figure 5.5: SSK BER Performance with RIS (K=2, N=16, J=2, M=2)

Combining all together, our properties still hold strong.

5.2 BER realistic scenario simulation

We can now simulate our framework in a realistic scenario. We first need to model the channel gain H and the path loss, based on the distance between the actors. We will define λ as the wavelenght of the signal, and d as the distance between two actors.

5.2.1 Channel gain calculation

We model the Rician fading [35] matrix $\boldsymbol{\Xi}$, to consider possible fading due to multipath interference. Using the Shape Parameter τ, defined as the ratio of the power contributions by line-of-sight path to the remaining multipaths, and the Scale parameter ξ, defined as the total power received in all paths, we can calculate

$$\nu^2 = \frac{\tau\xi}{1+\tau} \tag{5.1}$$

$$\sigma^2 = \frac{\xi}{2(1+\tau)} \tag{5.2}$$

and we can generate $\boldsymbol{\Xi}$ by creating a random complex matrix where the real and the imaginary values are extracted from a gaussian distribution $C(\frac{\nu}{\sqrt{2}}, \sigma)$ [34]

Then, given an actor r with n_r antennas disposed as a *uniform linear array*, we can define the *unit spatial signature in the directonal cosine* $\Omega = cos\phi$ [32] as

$$e_r(\Omega) = \frac{1}{\sqrt{n_r}} \begin{bmatrix} 1 \\ exp(-j2\pi\Delta\Omega) \\ exp(-j2\pi 2\Delta\Omega) \\ \vdots \\ exp(-j2\pi(n_r-1)\Delta\Omega) \end{bmatrix} \tag{5.3}$$

where

- Δ is the distance between the antennas (usually $\lambda/2$)
- ϕ is the angle of incidence of the line-of-sight onto the actor antenna

and we can model the channel gain matrix [32] as

$$\boldsymbol{H} = \boldsymbol{\Xi} \odot \sqrt{n_t n_r} exp(-j2\pi d/\lambda) e_r(\Omega_r) e_t(\Omega_t)^H \tag{5.4}$$

This equation can be used both for a direct transmission between two actors, or between an actor and a RIS.

5.2.2 Path loss calculation

We begin by modeling the free space path loss [33] between two points as

$$\boldsymbol{PL} = ((4\pi/\lambda)^2 d^k)^{-\frac{1}{2}} \tag{5.5}$$

where k is equal to 2 when the antennas are isontropic

For a direct LOS communication between the transmitter and another actor (either a legitimate receiver or an eavesdropper), the signal received from input x would be

$$y = \boldsymbol{PL}_B * \boldsymbol{B}x \tag{5.6}$$

Given a reflected signal with channel gain GPH, where

- $\boldsymbol{G}$ is the communication transmitter-RIS
- $\boldsymbol{H}$ is the communication RIS-actors
- $\boldsymbol{P}$ is the RIS reflection coefficient diagonal matrix

we have two different LOS communications. We have different way of calculating the total path loss:

- we consider the RISs to be active, meaning they amplify the signal received before reflecting it and so they negate the path loss reduction. The signal received would be $y = PL_H * GPHx$. In case of multiple RISs, only the last connection path loss is considered. We will call this as a *active path loss*
- we consider two separate path losses, one for each LOS. The signal received would be $y = PL_G * PL_H * \boldsymbol{GPH}x$. In case of multiple RISs, we multiply the path loss of all connections. We will call this as a *product path loss*
- we consider one single path loss from the sum of the two distances ($d = d_{t-RIS} + d_{RIS-r}$). The signal received would be $y = PL_{G+H} * \boldsymbol{GPH}x$. In case of multiple RISs, we add all the distances. We will call this as a *sum path loss*

We will see how these different considerations vary the results and the efficacy of the proposed framework.

- with *active path loss*, the RIS channel power is of the same order of magnitude as the transmitter channel power, so the direct signal receives significant noise
- with *product path loss*, the RIS channel power is orders of magnitude smaller. The message remains hidden in areas without direct line of sight to the transmitter
- with *sum path loss*, the disturbance effect is still visible, although less effective. It also the one with the results most similar to the theoretical simulation we made in the previous section.

The graphs below show some example situations, and prove our framework does also work in more realistic situations. Below, we used $\lambda = 0.08m, \tau = 0.6, \xi = 1, \eta = 0.9, SNR = 10db, K = 4\text{or}2, N = 16$. Each square represent an actor receiveing the signal (an eavesdropper, or a legitimate receiver if shown), with its own BER.

1 Ris, path loss: active

We can see here how this type of path loss ensures that in the vicinity of the RIS the Bit Error Rate remains very high, thanks to the active power component of the RIS itself.

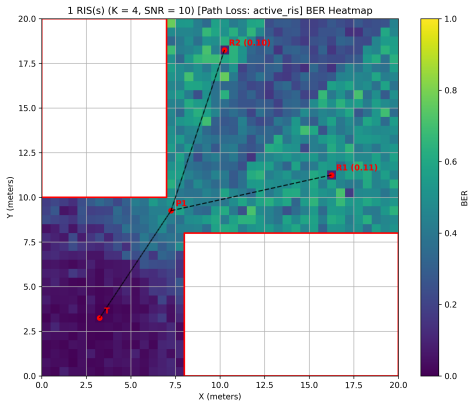

Figure 5.6: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: active ris] BER Heatmap

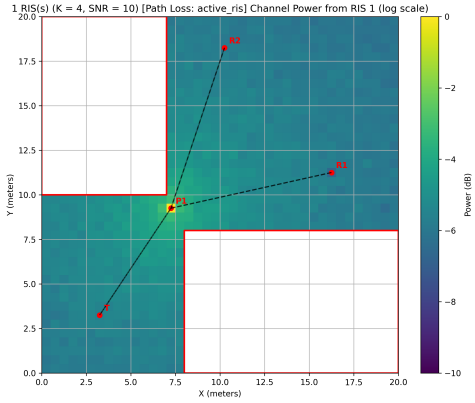

Figure 5.7: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: active ris] Channel Power from RIS 1 (log scale)

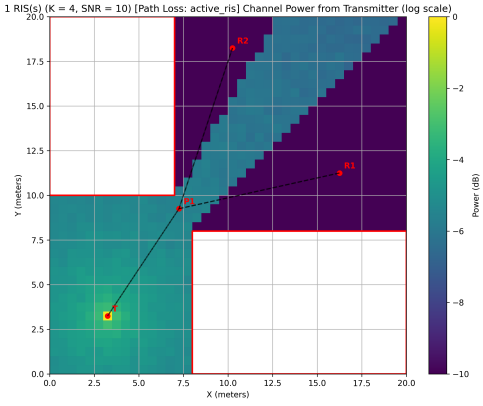

Figure 5.8: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: active ris] Channel Power from Transmitter (log scale)

1 Ris, path loss: product

With this type of path loss, the signal coming from the RIS has significant less power than the direct link from the transmitter, and cannot influence significantly the outcome. Outside LOS, the framework still works as expected.

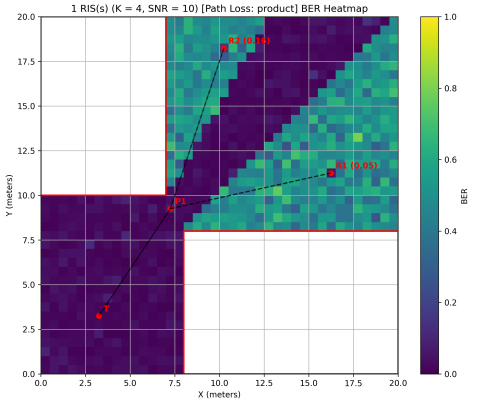

Figure 5.9: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: product] BER Heatmap

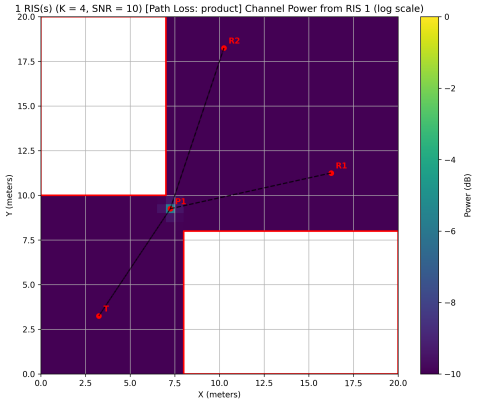

Figure 5.10: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: product] Channel Power from RIS 1 (log scale)

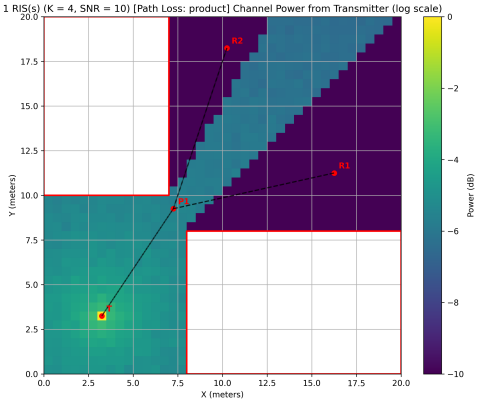

Figure 5.11: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: product] Channel Power from Transmitter (log scale)

1 Ris, path loss: sum

This is the path loss that better confirms our previous BER analysis. Without LOS, the BER for eavesdropper is stable at 0.5, the same as random guessing. With LOS, the RIS is still able to influence significantly the outcome, with more uniform noise that reduce the BER at 0.3

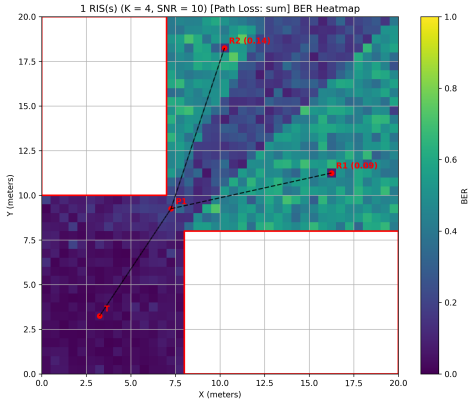

Figure 5.12: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: sum] BER Heatmap

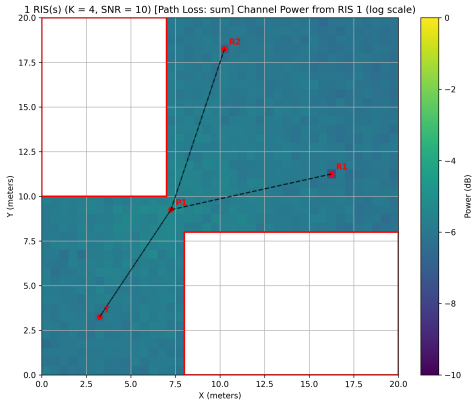

Figure 5.13: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: sum] Channel Power from RIS 1 (log scale)

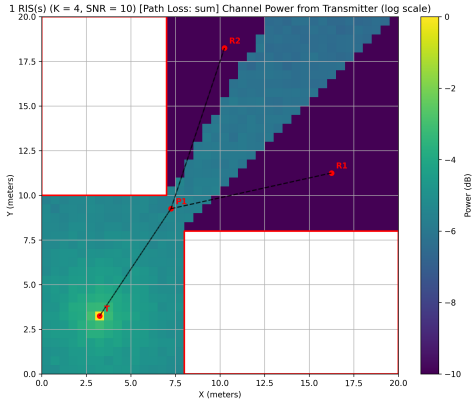

Figure 5.14: 1 RIS(s) (K = 4, SNR = 10) [Path Loss: sum] Channel Power from Transmitter (log scale)

2 Ris, path loss: active

Similar to the previous scenario, we can clearly see where the RIS have significant power to influence the signal reception, and where without LOS the signal is undeciphrable for eavesdroppers.

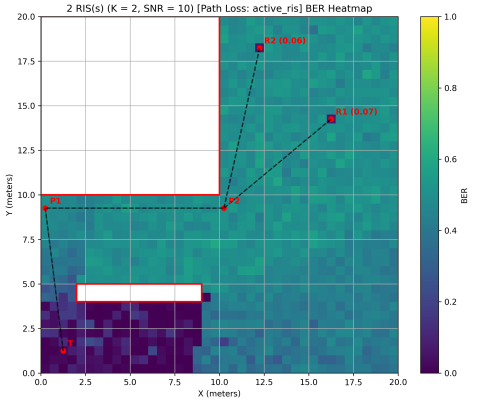

Figure 5.15: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: active ris] BER Heatmap

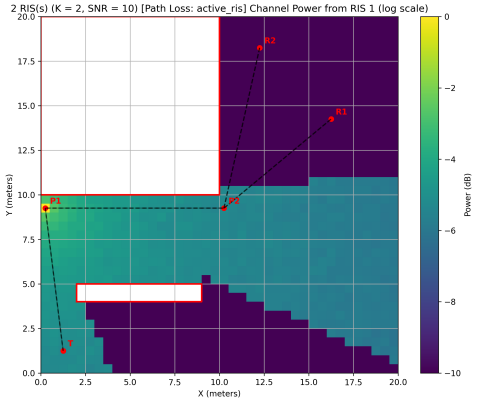

Figure 5.16: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: active ris] Channel Power from RIS 1 (log scale)

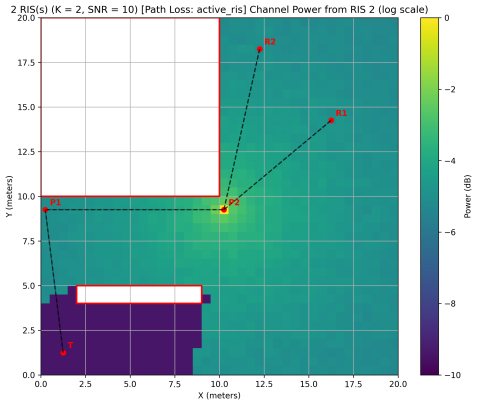

Figure 5.17: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: active ris] Channel Power from RIS 2 (log scale)

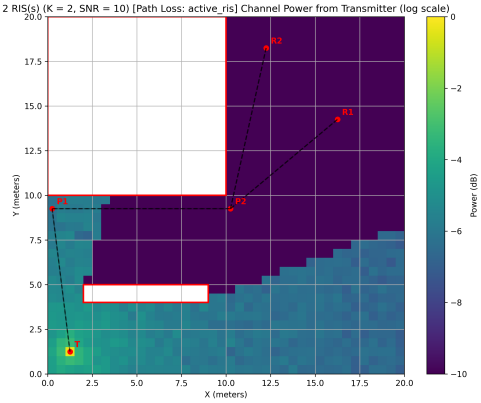

Figure 5.18: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: active ris] Channel Power from Transmitter (log scale)

2 Ris, path loss: product

We can draw similar conclusions as before for the product path loss.

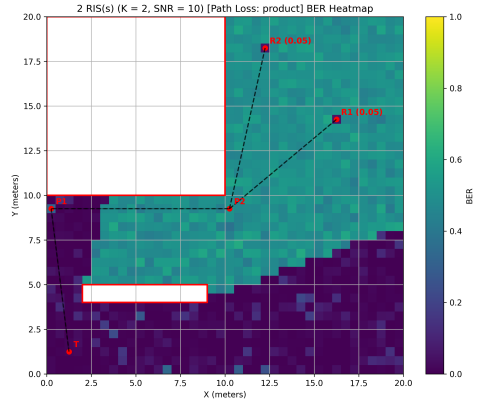

Figure 5.19: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: product] BER Heatmap

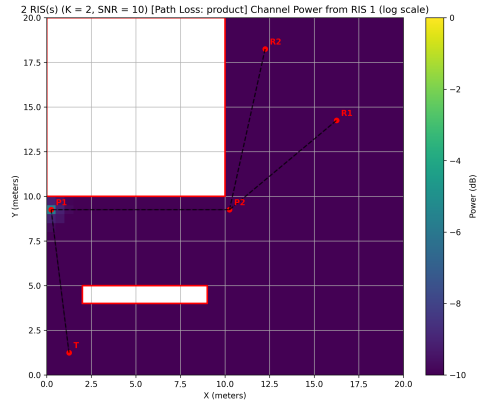

Figure 5.20: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: product] Channel Power from RIS 1 (log scale)

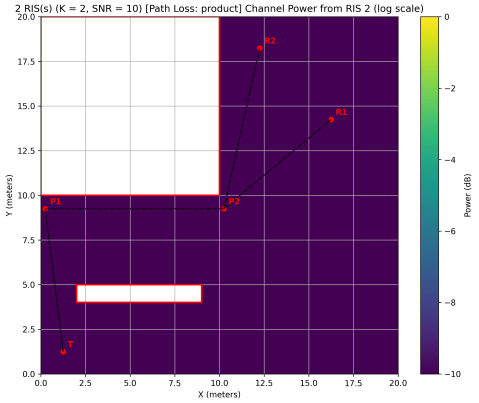

Figure 5.21: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: product] Channel Power from RIS 2 (log scale)

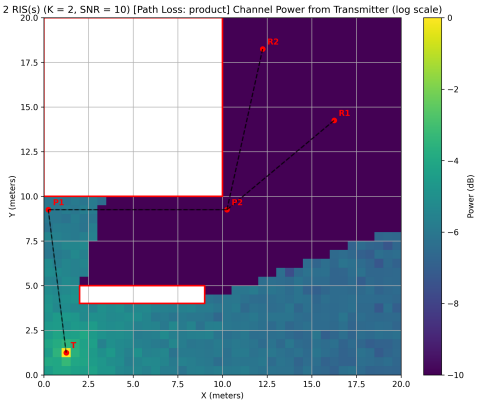

Figure 5.22: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: product] Channel Power from Transmitter (log scale)

2 Ris, path loss: sum

The distinction between having or not LOS are very visible here. The displayed values for the BER are in line with the previous graph on the same kind of path loss.

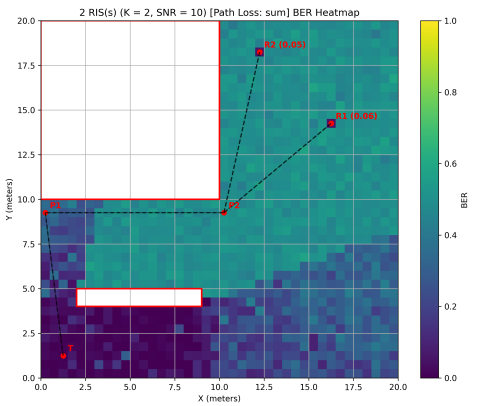

Figure 5.23: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: sum] BER Heatmap

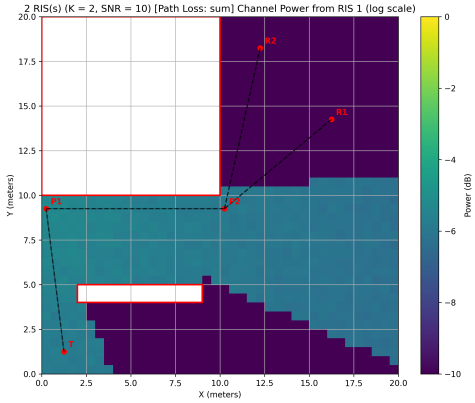

Figure 5.24: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: sum] Channel Power from RIS 1 (log scale)

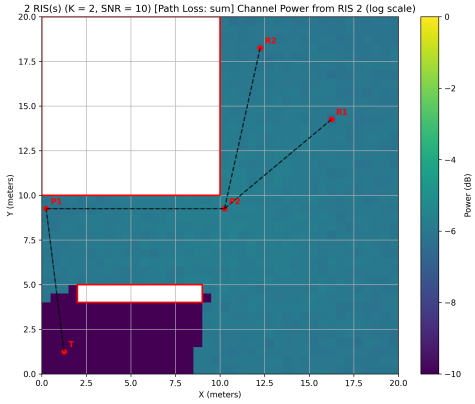

Figure 5.25: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: sum] Channel Power from RIS 2 (log scale)

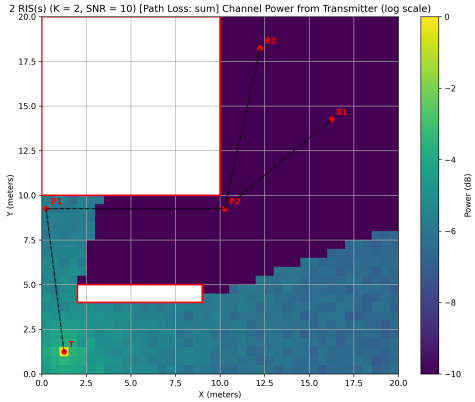

Figure 5.26: 2 RIS(s) (K = 2, SNR = 10) [Path Loss: sum] Channel Power from Transmitter (log scale)

6 Code Implementation

To validate our theoretical framework, we have implemented a simulation environment in Python that allows us to test both our mathematical models and their real-world applicability. The codebase consists of three main modules: diagonalization, bit error rate (BER) analysis, and a heatmap generator to visualize the spatial distribution of signal quality.

6.1 Diagonalization Module

The diagonalization module implements the core mathematical concepts of our RIS reflection framework, following the theoretical foundation presented in Section 3 and 4.

6.1.1 Null Space Calculation

We first start by making the calculation of reflection matrices, which ensure the effective channel between the transmitter and receiver is diagonalized. We implement this through the `calculate_W_single` and `calculate_W_multiple` functions:

```python
def calculate_W_single(K: int, N: int, G: np.ndarray, H: np.ndarray) -> np.ndarray:
    """
    Calculate W matrix for a single receiver.

    Args:
        K: Number of antennas
        N: Number of reflecting elements
        G: Channel matrix from RIS to receiver (KxN)
        H: Channel matrix from transmitter to RIS (NxK)

    Returns:
        W: The W matrix as defined in equation (8) of the paper
    """
    W = np.zeros((N, N), dtype=complex)

    for i in range(K):
        for j in range(K):
            if i != j:
                temp = np.multiply(G[j, :], H[:, i].T)
                W += np.outer(temp.conj(), temp)

    return W
```

Code 6.1: Calculation of W matrices

This function calculates the W matrix for a single receiver. For multiple receivers, we stack these matrices:

```python
def calculate_W_multiple(K: int, N: int, J: int, Gs: List[np.ndarray], H: np.
    ndarray) -> np.ndarray:
    """
    Calculate combined W matrix for multiple receivers.
    """
    W_combined = np.zeros((J * N, N), dtype=complex)

    for j in range(J):
        W_j = calculate_W_single(K, N, Gs[j], H)
        W_combined[j*N:(j+1)*N, :] = W_j

    return W_combined
```

Code 6.2: W matrix for multiple receivers

6.1.2 RIS Reflection Matrix Calculation

With the W matrix defined, we can calculate the reflection matrices using Singular Value Decomposition (SVD) to find the null space:

```python
def calculate_ris_reflection_matrice(
    K: int,
    N: int,
    J: int,
    Gs: List[np.ndarray],
    H: np.ndarray,
    eta: float,
) -> Tuple[np.ndarray, float]:
    """
    Calculate reflection matrices for RIS surfaces.
    """
    W = calculate_W_multiple(K, N, J, Gs, H)
    U, sigma, Vh = np.linalg.svd(W)

    null_space_dim = N - J*K**2 + J*K
    if null_space_dim <= 0:
        raise ValueError(f"No solution exists. Need more reflecting elements.")

    first_singular_values = sigma[:N - null_space_dim]
    last_singular_values = sigma[-null_space_dim:]

    null_space_basis = Vh[-null_space_dim:, :].T.conj()

    if null_space_basis.shape != (N, null_space_dim):
        raise ValueError(f"Invalid null space basis shape.")

    a = np.random.normal(0, 1, (null_space_dim,)) + 1j * np.random.normal(0, 1, (null_space_dim,))

    p_unnormalized = null_space_basis @ a
    p = eta * p_unnormalized / np.max(np.abs(p_unnormalized))
    P = np.diag(p)
    dor = 2 * null_space_dim
    return P, dor
```

Code 6.3: RIS Reflection Matrix Calculation

This function implements the theoretical approach described in Section 4.1, where we find the null space of W and then generate a random vector in this space to ensure randomness in our reflection coefficients.

6.1.3 Multiple RIS Support

To support multiple RIS surfaces in series, we implemented:

```python
def calculate_multi_ris_reflection_matrices(
    K: int,
    N: int,
    J: int,
    M: int,
    Gs: List[np.ndarray],
    H: np.ndarray,
    eta: float,
    Cs: List[np.ndarray]
) -> Tuple[List[np.ndarray], float]:
    """
    Calculate reflection matrices for M RIS surfaces.
    """
    if len(Cs) != M-1:
        raise ValueError(f"Expected {M-1} inter-RIS channel matrices.")

    ps = []
    S = np.eye(N)
```

```
    # Generate M-1 random reflection vectors
    for i in range(M-1):
        absorptions = np.random.uniform(0, 1, N)
        phases = np.random.uniform(0, 2*np.pi, N)
        # p_m[i] = eta * r_i * exp(j*theta_i)
        p_m = absorptions * np.exp(1j * phases)
        ps.append(p_m)
        S = S @ np.diag(p_m) @ Cs[i]

    Gs_prime = [G @ S for G in Gs]

    # Calculate the last reflection matrix
    P_final, dor = calculate_ris_reflection_matrice(K, N, J, Gs_prime, H, eta)
    p_final = np.diag(P_final)
    ps.append(p_final)

    Ps = []
    for pm in ps:
        Ps.append(np.diag(pm))

    return Ps, dor
```

Code 6.4: Multiple RIS calculation

This implements our approach from Section 4.3, where we generate random reflection coefficients for all but the last RIS, then calculate the last one to ensure diagonalization.

6.1.4 Unified Reflection Matrix

For convenience in calculations, we provide a function to unify multiple reflection matrices into a single effective matrix:

```
def unify_ris_reflection_matrices(Ps: List[np.ndarray], Cs: List[np.ndarray]) -> np.ndarray:
    """
    Unify reflection matrices into a single matrix.
    """
    P = Ps[0]
    for i in range(len(Ps)-1):
        P = P @ Cs[i] @ Ps[i+1]
    return P
```

Code 6.5: Unifying RIS matrices

6.1.5 Verification

We also implement verification functions to ensure our theoretical expectations match the implementation:

```
def verify_multi_ris_diagonalization(
    Ps: List[np.ndarray],
    Gs: List[np.ndarray],
    H: np.ndarray,
    Cs: List[np.ndarray]
) -> List[bool]:
    """
    Verify that G(P_1C_1P_2C_2...P_M)H is diagonal for all receivers.
    """
    results = []

    P = unify_ris_reflection_matrices(Ps, Cs)
    for G in Gs:
        effective_channel = G @ P @ H
        off_diag_sum = np.sum(np.abs(effective_channel - np.diag(np.diag(effective_channel))))
        results.append(off_diag_sum < tolerance)
```

```
    return results
```

Code 6.6: Verification of diagonalization

6.2 BER Module

The BER (Bit Error Rate) module implements the simulation of space shift keying (SSK) transmissions and calculates the error rates under different scenarios.

6.2.1 SSK Transmission Simulation

We simulate SSK transmission with the following core functions:

```
def simulate_ssk_transmission(K: int, sigma_sq: float, calculate_detected_id:
    Callable[[np.ndarray, np.ndarray], float]):
    n_bits = int(np.log2(K))
    if 2**n_bits != K:
        raise ValueError(f"K must be a power of 2, got {K}")
    bit_mappings = np.array([format(i, f'0{n_bits}b') for i in range(K)])
    true_bits = np.random.randint(0, 2, n_bits)
    true_bits_str = ''.join(map(str, true_bits))
    true_idx = np.where(bit_mappings == true_bits_str)[0][0]

    x = np.zeros(K)
    x[true_idx] = 1

    noise = create_random_noise_vector(K, sigma_sq)
    detected_idx = calculate_detected_id(x, noise)

    detected_bits = np.array(list(bit_mappings[detected_idx])).astype(int)
    errors = np.sum(detected_bits != true_bits)
    return errors / n_bits
```

Code 6.7: SSK Transmission Simulation

This function implements the common core of our SSK transmission simulation. It maps bits to antenna indices, simulates transmission with noise, and calculates the error rate.

We then implement specialized versions for reflection and direct transmission:

```
def simulate_ssk_transmission_reflection(K: int, effective_channel: np.ndarray,
    sigma_sq: float):
    if effective_channel.shape != (K, K):
        raise ValueError(f"Reflection: Effective channel shape must be ({K}, {K})")

    def calculate_detected_id(x: np.ndarray, noise: np.ndarray):
        y = effective_channel @ x + noise
        return np.argmax(np.abs(y)**2)

    return simulate_ssk_transmission(K, sigma_sq, calculate_detected_id)
```

Code 6.8: SSK Transmission with Reflection

```
def simulate_ssk_transmission_direct(K: int, B: np.ndarray, effective_channel: np.
    ndarray, sigma_sq: float):
    if B.shape != (K, K):
        raise ValueError(f"Direct: B shape must be ({K}, {K})")

    if effective_channel.shape != (K, K):
        raise ValueError(f"Direct: Effective channel shape must be ({K}, {K})")

    def calculate_detected_id(x: np.ndarray, noise: np.ndarray):
        y = (B + effective_channel) @ x + noise
        distances = np.array([np.linalg.norm(y - B[:, i]) for i in range(B.shape
    [1])])
        return np.argmin(distances)

    return simulate_ssk_transmission(K, sigma_sq, calculate_detected_id)
```

Code 6.9: SSK Transmission with Direct Path

These functions implement the detection methods described in Section 3.2, where the legitimate receiver can detect the signal through the diagonalized channel, while the eavesdropper hears from the direct path and receives interference from the reflection.

6.2.2 BER Simulation

We also implement a comprehensive BER simulation function that evaluates performance across different SNR values:

```
def calculate_ber_simulation(snr_db, K, N, J, M, eta=0.9, num_symbols=10000):
    sigma_sq = snr_db_to_sigma_sq(snr_db)
    errors_receiver = 0
    errors_eavesdropper = 0
    errors_direct = 0

    errors_receiver_double = 0
    errors_eavesdropper_double = 0

    for _ in range(num_symbols):
        # Generate channel matrices
        H = generate_random_channel_matrix(N, K)
        Gs = [generate_random_channel_matrix(K, N) for _ in range(J)]
        G = random.choice(Gs)
        Fs = [generate_random_channel_matrix(K, N) for _ in range(M)]
        B = generate_random_channel_matrix(K, K)
        Cs = [generate_random_channel_matrix(N, N) for _ in range(M-1)]

        # Calculate reflection matrices
        Ps, _ = calculate_multi_ris_reflection_matrices(K, N, J, M, Gs, H, eta, Cs)
        P = unify_ris_reflection_matrices(Ps, Cs)

        # Calculate effective channels
        effective_channel_receiver = G @ P @ H
        effective_channel_eavesdropper = np.zeros((K, K), dtype=np.complex128)
        for i in range(M):
            P_to_i = unify_ris_reflection_matrices(Ps[:i+1], Cs[:i])
            effective_channel_eavesdropper += Fs[i] @ P_to_i @ H
        effective_channel_direct = np.zeros((K, K))

        # Simulate transmissions
        errors_receiver += simulate_ssk_transmission_reflection(K,
    effective_channel_receiver, sigma_sq)
        errors_eavesdropper += simulate_ssk_transmission_direct(K, B,
    effective_channel_eavesdropper, sigma_sq)
        errors_direct += simulate_ssk_transmission_direct(K, B,
    effective_channel_direct, sigma_sq)

        H2 = generate_random_channel_matrix(N, K)
        Gs2 = [generate_random_channel_matrix(K, N) for _ in range(J)]
        G2 = random.choice(Gs2)
        Fs2 = [generate_random_channel_matrix(K, N) for _ in range(M)]
        Cs2 = [generate_random_channel_matrix(N, N) for _ in range(M-1)]
        Ps2, _ = calculate_multi_ris_reflection_matrices(
            K, N, J, M, Gs2, H2, eta, Cs2
        )
        P2 = unify_ris_reflection_matrices(Ps2, Cs2)

        effective_channel_receiver_2 = G2 @ P2 @ H2
        effective_channel_eavesdropper_2 = np.zeros((K, K), dtype=np.complex128) #
    F @ P @ H
        for i in range(M):
            P_to_i = unify_ris_reflection_matrices(Ps2[:i+1], Cs2[:i])
            effective_channel_eavesdropper_2 += Fs2[i] @ P_to_i @ H2

        effective_channel_receiver_double = effective_channel_receiver +
    effective_channel_receiver_2
```

```
        effective_channel_eavesdropper_double = effective_channel_eavesdropper +
    effective_channel_eavesdropper_2

        errors_receiver_double += simulate_ssk_transmission_reflection(K,
    effective_channel_receiver_double, sigma_sq)
        errors_eavesdropper_double += simulate_ssk_transmission_direct(K, B,
    effective_channel_eavesdropper_double, sigma_sq)

    result_receiver = errors_receiver / num_symbols
    result_eavesdropper = errors_eavesdropper / num_symbols
    result_direct = errors_direct / num_symbols
    result_receiver_double = errors_receiver_double / num_symbols
    result_eavesdropper_double = errors_eavesdropper_double / num_symbols

    return result_receiver, result_eavesdropper, result_direct,
    result_receiver_double, result_eavesdropper_double
```

Code 6.10: BER Simulation

This function simulates BER performance across legitimate receivers and eavesdroppers, including scenarios with multiple RIS surfaces and different path configurations.

6.3 Heatmap Generator

To visualize our results in a spatial context, we implemented a heatmap generator that simulates signal quality across a 2D space:

6.3.1 Core Heatmap Class

```
class HeatmapGenerator:
    def __init__(self, width: int, height: int, resolution: float = 0.5):
        """
        Initialize the heatmap generator with given dimensions.
        """
        self.width = width
        self.height = height
        self.resolution = resolution

        # Calculate grid dimensions based on resolution
        self.grid_width = int(width / resolution)
        self.grid_height = int(height / resolution)
        self.grid = np.zeros((self.grid_height, self.grid_width))
        self.buildings = []
        # Dictionary to store points with their labels and coordinates
        self.points = {}
```

Code 6.11: Heatmap Generator Class

The heatmap generator creates a grid-based representation of a physical space, where we can place transmitters, receivers, and obstacles.

6.3.2 Building and Point Management

```
def add_building(self, x: int, y: int, width: int, height: int):
    """
    Add a building to the map. Buildings are excluded from the heatmap calculation.
    """
    self.buildings.append((x, y, width, height))
    grid_x, grid_y = self._meters_to_grid(x, y)
    grid_width = int(width / self.resolution)
    grid_height = int(height / self.resolution)

    # Mark building area as NaN to exclude from heatmap
    self.grid[grid_y:grid_y+grid_height, grid_x:grid_x+grid_width] = np.nan

def add_point(self, label: str, x: float, y: float):
    """
```

```
    Add a point of interest to the map with a specific label.
    """
    if not (0 <= x < self.width and 0 <= y < self.height):
        raise ValueError(f"Point {label} coordinates ({x}, {y}) are outside the map boundaries")
    self.points[label] = (x, y)
```

Code 6.12: Buildings and Points in Heatmap

These functions allow us to define the simulation environment with buildings (which block signals) and points representing transmitters, RIS surfaces, and receivers.

6.3.3 Line of Sight Checking

An important aspect of our simulation is determining whether two points have line-of-sight:

```
def _line_intersects_building(self, x1: float, y1: float, x2: float, y2: float) -> bool:
    """
    Check if line between two points intersects any building.
    Uses line segment intersection algorithm.
    """
    def ccw(A: tuple, B: tuple, C: tuple) -> bool:
        """Returns True if points are counter-clockwise oriented"""
        return (C[1] - A[1]) * (B[0] - A[0]) > (B[1] - A[1]) * (C[0] - A[0])

    def intersect(A: tuple, B: tuple, C: tuple, D: tuple) -> bool:
        """Returns True if line segments AB and CD intersect"""
        return ccw(A, C, D) != ccw(B, C, D) and ccw(A, B, C) != ccw(A, B, D)

    for bx, by, bw, bh in self.buildings:
        building_corners = [
            (bx, by), (bx + bw, by),
            (bx + bw, by + bh), (bx, by + bh)
        ]

        for i in range(4):
            if intersect(
                (x1, y1), (x2, y2),
                building_corners[i], building_corners[(i + 1) % 4]
            ):
                return True
    return False
```

Code 6.13: Line of Sight Checking

This function allows us to determine if buildings block the line-of-sight between points, which is crucial for accurate path loss simulation.

6.3.4 Distance Calculation

To model path loss, we calculate distances between points:

```
def calculate_distance_from_point(self, point: str) -> np.ndarray:
    """
    Calculate the minimum distance from each grid cell to the specified point.
    """
    distances = np.full_like(self.grid, np.inf)
    px, py = self.points[point]

    for grid_y in range(self.grid_height):
        for grid_x in range(self.grid_width):
            if np.isnan(self.grid[grid_y, grid_x]):
                continue

            x, y = self._grid_to_meters(grid_x, grid_y)
            if self._line_intersects_building(x, y, px, py):
                continue

```

```
            distance = np.sqrt((x - px)**2 + (y - py)**2)
            distances[grid_y, grid_x] = distance

    return distances
```

Code 6.14: Distance Calculation

This function creates a grid where each cell contains the distance to a specified point, setting infinite distance for points without line-of-sight due to buildings.

6.3.5 Channel Model Functions

We implement realistic channel models for our simulations:

```
def calculate_free_space_path_loss(d: float, lam = 0.08, k = 2) -> float:
    """
    Calculate free space path loss between transmitter and receiver
    """
    if d == 0: d = 0.01
    return 1 / np.sqrt((4 * np.pi / lam) ** 2 * d ** k)

def calculate_unit_spatial_signature(incidence: float, K: int, delta: float):
    """
    Calculate the unit spatial signature vector for a given angle of incidence
    """
    directional_cosine = np.cos(incidence)
    e = np.array([(1 / np.sqrt(K)) * np.exp(-1j * 2 * np.pi * (k - 1) * delta *
    directional_cosine) for k in range(K)])
    return e.reshape(-1, 1)

def generate_rice_faiding_channel(L: int, K: int, ratio: float, total_power = 1.0)
    -> np.ndarray:
    """
    Generate a Ricean fading channel matrix
    """
    nu = np.sqrt(ratio * total_power / (1 + ratio))
    sigma = np.sqrt(total_power / (2 * (1 + ratio)))
    return generate_rice_matrix(L, K, nu, sigma)

def calculate_mimo_channel_gain(d: float, L: int, K: int, lam = 0.08, k = 2) ->
    tuple[np.ndarray, float]:
    """
    Calculate MIMO channel gains between transmitter and receiver
    """
    if d == np.inf:
        return np.zeros((K, L), dtype=complex)
    if d == 0:
        d = 0.5

    delta = lam / 2
    c = np.sqrt(L * K) * np.exp(-1j * 2 * np.pi * d / lam)
    e_r = calculate_unit_spatial_signature(0, K, delta)
    e_t = calculate_unit_spatial_signature(0, L, delta)
    H = c * (e_r @ e_t.T.conj())

    ratio = 0.6
    total_power = 1.0
    H = H * generate_rice_faiding_channel(L, K, ratio, total_power)
    return H
```

Code 6.15: Channel Modeling Functions

These functions implement the channel models described in Section 5.2, including free space path loss, spatial signatures, and Ricean fading.

6.3.6 BER Heatmap Simulation

Finally, we put everything together in a function that simulates BER across the entire space:

```python
def ber_heatmap_reflection_simulation(
    width: int,
    height: int,
    buildings: List[Tuple[int, int, int, int]],
    transmitter: Tuple[int, int],
    ris_points: List[Tuple[int, int]],
    receivers: List[Tuple[int, int]],
    num_symbols: int,
    N: int = 16,
    K: int = 2,
    eta: float = 0.9,
    snr_db: int = 10,
    path_loss_calculation_type: Literal['sum', 'product', 'active_ris'] = 'sum'
):
    """
    Run RIS reflection simulation with given parameters
    """
    ber_heatmap = HeatmapGenerator(width, height)

    # Setup environment
    for building in buildings:
        ber_heatmap.add_building(*building)

    tx, ty = transmitter
    ber_heatmap.add_point('T', tx, ty)

    M = len(ris_points)
    for i, (px, py) in enumerate(ris_points):
        ber_heatmap.add_point(f'P{i+1}', px, py)

    J = len(receivers)
    for i, (rx, ry) in enumerate(receivers):
        ber_heatmap.add_point(f'R{i+1}', rx, ry)

    # Calculate distance matrices
    distances_from_T = ber_heatmap.calculate_distance_from_point('T')
    distances_from_Ps = [ber_heatmap.calculate_distance_from_point(f'P{i+1}') for i
     in range(M)]

    # Create heatmaps for power analysis
    power_heatmap_from_T = HeatmapGenerator.copy_from(ber_heatmap)
    power_heatmap_from_Ps = [HeatmapGenerator.copy_from(ber_heatmap) for _ in range
    (M)]

    # Calculate channel matrices
    tx_grid_y, tx_grid_x = ber_heatmap._meters_to_grid(tx, ty)
    H = calculate_mimo_channel_gain(distances_from_Ps[0][tx_grid_y, tx_grid_x], K,
    N)

     if M > 1:
        receiver_grid_coords = [(ber_heatmap._meters_to_grid(rx, ry)) for rx, ry in
     receivers]
        Gs = [calculate_mimo_channel_gain(distances_from_Ps[-1][ry, rx], N, K)
              for ry, rx in receiver_grid_coords]

        ris_grid_coords = [ber_heatmap._meters_to_grid(px, py) for px, py in
    ris_points]
        Cs = [calculate_mimo_channel_gain(
            distances_from_Ps[i+1][ris_grid_coords[i][1], ris_grid_coords[i][0]],
            N, N
        ) for i in range(M-1)]
    else:
        receiver_grid_coords = [(ber_heatmap._meters_to_grid(rx, ry)) for rx, ry in
     receivers]
        Gs = [calculate_mimo_channel_gain(distances_from_Ps[0][ry, rx], N, K)
              for ry, rx in receiver_grid_coords]
```

```
        Cs = []

    print(f"Channel matrix from transmitter to RIS: Power {calculate_channel_power(H):.1e}")
    print(f"Channel matrix from RIS to receiver: Power {calculate_channel_power(Gs[0]):.1e}")

    Ps, _ = calculate_multi_ris_reflection_matrices(K, N, J, M, Gs, H, eta, Cs)
    P = unify_ris_reflection_matrices(Ps, Cs)
    print(f"Reflection matrix: Power {calculate_channel_power(P):.1e}")
    print(f"Effective channel matrix: Power {calculate_channel_power(Gs[0] @ P @ H):.1e}")
    print()

    # * Calculate cumulative path distances
    ris_path_distances = []
    for i in range(M):
        if i == 0:
            # * Distance from T to first RIS
            ris_path_distances.append(distances_from_Ps[0][ty, tx])
        else:
            # * Distance between consecutive RIS points
            ris_path_distances.append(
                distances_from_Ps[i][ris_points[i-1][1], ris_points[i-1][0]]
            )

    # Define BER calculation function for each point
    def calculate_ber_per_point(x: int, y: int) -> float:
        grid_x, grid_y = ber_heatmap._meters_to_grid(x, y)
        distance_from_T = distances_from_T[grid_y, grid_x]
        B = calculate_mimo_channel_gain(distance_from_T, K, K) * calculate_free_space_path_loss(distance_from_T)
        B_power = calculate_channel_power(B)
        power_heatmap_from_T.grid[grid_y, grid_x] = B_power

        distances_from_Ps_current = [distances_from_Ps[i][grid_y, grid_x] for i in range(M)]
        Fs = [calculate_mimo_channel_gain(d, N, K) for d in distances_from_Ps_current]

        # * Override channel matrices for receiver positions
        for j in range(J):
            if x == receivers[j][0] and y == receivers[j][1]:
                Fs[-1] = Gs[j]

        errors = 0
        for _ in range(num_symbols):
            Ps, _ = calculate_multi_ris_reflection_matrices(K, N, J, M, Gs, H, eta, Cs)
            P = unify_ris_reflection_matrices(Ps, Cs)

            effective_channel = np.zeros((K, K), dtype=complex)

            # Handle different path loss types
            for i in range(M):
                if i == 0:
                    P_to_i = Ps[0]
                else:
                    P_to_i = unify_ris_reflection_matrices(Ps[:i+1], Cs[:i])

                if path_loss_calculation_type == 'sum':
                    total_distance = sum(ris_path_distances[:i+1]) + distances_from_Ps_current[i]
                    total_path_loss = calculate_free_space_path_loss(total_distance)
                    new_effective_channel = Fs[i] @ P_to_i @ H * total_path_loss
```

```
                elif path_loss_calculation_type == 'product':
                    total_path_loss = 1
                    for j in range(i+1):
                        total_path_loss *= calculate_free_space_path_loss(ris_path_distances[j])
                    total_path_loss *= calculate_free_space_path_loss(distances_from_Ps_current[i])
                    new_effective_channel = Fs[i] @ P_to_i @ H * total_path_loss
                elif path_loss_calculation_type == 'active_ris':
                    total_path_loss = calculate_free_space_path_loss(distances_from_Ps_current[i])
                    new_effective_channel = Fs[i] @ P_to_i @ H * total_path_loss
                else:
                    raise ValueError(f"Invalid path loss calculation type: {path_loss_calculation_type}")

                effective_channel += new_effective_channel

            # Determine signal power and calculate BER
            power = B_power if distance_from_T != np.inf else calculate_channel_power(effective_channel)
            sigma_sq = snr_db_to_sigma_sq(snr_db, power)

            if distance_from_T == np.inf:
                errors += simulate_ssk_transmission_reflection(K, effective_channel, sigma_sq)
            else:
                errors += simulate_ssk_transmission_direct(K, B, effective_channel, sigma_sq)

        return errors / num_symbols

    # Apply BER calculation to each point in the grid
    ber_heatmap.apply_function(calculate_ber_per_point)

    # Visualize results
    title = f'{M} RIS(s) (K = {K}, SNR = {snr_db}) [Path Loss: {path_loss_calculation_type}]'
    ber_heatmap.visualize(title + ' BER Heatmap', vmin=0.0, vmax=1.0, label='BER', show_receivers_values=True)
    ber_heatmap.visualize(title + ' BER Heatmap', log_scale=True, vmin=-10.0, vmax=0.0, label='BER', show_receivers_values=True)

    power_heatmap_from_T.visualize(title + ' Channel Power from Transmitter', log_scale=True, vmin=-10.0, vmax=0.0, label='Power (dB)')
    for i in range(M):
        power_heatmap_from_Ps[i].visualize(title + f' Channel Power from RIS {i+1}', log_scale=True, vmin=-10.0, vmax=0.0, label='Power (dB)')
```

Code 6.16: BER Heatmap Simulation

This function implements our full simulation, calculating BER at each point in the space based on our theoretical framework and the different path loss models discussed in Section 5.2.

6.4 Main Simulation Scenarios

The main function in the heatmap.py file demonstrates how we use these components to evaluate different scenarios:

```
def main():
    # One reflection simulation
    buildings_single = [
        (0, 10, 7, 10),
        (8, 0, 12, 8)
    ]
    transmitter_single = (3, 3)
```

```
    ris_points_single = [(7, 9)]
    receivers_single = [(16, 11), (10, 18)]

    for path_loss_calculation_type in PATH_LOSS_TYPES:
        ber_heatmap_reflection_simulation(
            width=20,
            height=20,
            buildings=buildings_single,
            transmitter=transmitter_single,
            ris_points=ris_points_single,
            receivers=receivers_single,
            N=25,
            K=4,
            path_loss_calculation_type=path_loss_calculation_type,
            num_symbols=num_symbols
        )

    # Multiple reflection simulation
    buildings_multiple = [
        (0, 10, 10, 10),
        (2, 4, 7, 1)
    ]
    transmitter_multiple = (1, 1)
    ris_points_multiple = [(0, 9), (10, 9)]
    receivers_multiple = [(16, 14), (12, 18)]

    for path_loss_calculation_type in PATH_LOSS_TYPES:
        ber_heatmap_reflection_simulation(
            width=20,
            height=20,
            buildings=buildings_multiple,
            transmitter=transmitter_multiple,
            ris_points=ris_points_multiple,
            receivers=receivers_multiple,
            N=16,
            K=2,
            path_loss_calculation_type=path_loss_calculation_type,
            num_symbols=num_symbols
        )
```

Code 6.17: Main Simulation Scenarios

This function sets up and runs simulations for two distinct scenarios: 1. A single RIS reflection setup with two buildings, a transmitter, and two receivers 2. A multiple RIS reflection setup with two RIS surfaces in series

Each scenario is simulated with all three path loss models: sum, product, and active RIS. These simulations generate the heatmaps presented in Section 5.2, allowing us to visualize how BER varies across space under different conditions.

The simulations for the BER plots shown in section 5.1 are implemented in the ‘plot_ber_curves()‘ function:

```
def plot_ber_curves():
    N = 16    # Number of reflecting elements
    K = 2     # Number of antennas
    eta = 0.9 # Reflection efficiency

    for J in range(1, 3):  # Number of receivers
        for M in range(1, 3):  # Number of RIS surfaces
            print(f"Processing J={J}, M={M}")
            snr_range_db = np.arange(-10, 31, 2)
            ber_simulated_receiver = []
            ber_simulated_eavesdropper = []
            ber_simulated_direct = []
            ber_simulated_receiver_double = []
            ber_simulated_eavesdropper_double = []

```

```
            for snr_db in snr_range_db:
                result_receiver, result_eavesdropper, result_direct,
    result_receiver_double, result_eavesdropper_double = calculate_ber_simulation(
    snr_db, K, N, J, M, eta)
                ber_simulated_receiver.append(result_receiver)
                ber_simulated_eavesdropper.append(result_eavesdropper)
                ber_simulated_direct.append(result_direct)
                ber_simulated_receiver_double.append(result_receiver_double)
                ber_simulated_eavesdropper_double.append(result_eavesdropper_double
    )
                print(f"Processed SNR = {snr_db} dB:\t{result_receiver:.2f}\t{
    result_eavesdropper:.2f}\t{result_direct:.2f}")

            plt_name = f'SSK BER Performance with RIS (K={K}, N={N}, J={J}, M={M})'
            plt.figure(figsize=(10, 6))
            plt.semilogy(snr_range_db, ber_simulated_direct, label=f'Simulation
    Direct')
            plt.semilogy(snr_range_db, ber_simulated_receiver, label='Simulation
    Receiver')
            plt.semilogy(snr_range_db, ber_simulated_receiver_double, label='
    Simulation Receiver Double RIS Source')
            plt.semilogy(snr_range_db, ber_simulated_eavesdropper, label=f'
    Simulation Eavesdropper')
            plt.semilogy(snr_range_db, ber_simulated_eavesdropper_double, label=f'
    Simulation Eavesdropper Double RIS Source')
            plt.grid(True)
            plt.xlabel('SNR (dB)')
            plt.ylabel('Bit Error Rate (BER)')
            plt.title(plt_name)
            plt.legend()
            plt.savefig(f"./simulations/results/{plt_name}.png", dpi=300, format='
    png')
            print(f"Saved {plt_name}.png\n\n")
```

Code 6.18: BER Curve Plotting Function

This function systematically evaluates BER performance across different signal-to-noise ratios (SNR), generating plots for each combination of receiver count (J) and RIS surface count (M). These plots allow us to compare the performance of legitimate receivers versus eavesdroppers under different conditions.

6.5 Utils Module

Although not shown in the provided code snippets, we also implemented a util module that handles noise generation and SNR calculations:

```
def snr_db_to_sigma_sq(snr_db: float, power: float = 1.0) -> float:
    """
    Convert SNR in dB to noise variance

    Parameters:
    -----------
    snr_db : SNR in dB
    power : Signal power (default 1.0)

    Returns:
    --------
    Noise variance sigma_sq
    """
    snr_linear = 10**(snr_db/10)
    return power / snr_linear

def create_random_noise_vector(size: int, sigma_sq: float) -> np.ndarray:
    """
    Create a random noise vector with specified variance

```

```
    Parameters:
    -----------
    size : Size of the noise vector
    sigma_sq : Noise variance

    Returns:
    --------
    Random noise vector
    """
    return np.random.normal(0, np.sqrt(sigma_sq/2), (size,)) + 1j * np.random.
    normal(0, np.sqrt(sigma_sq/2), (size,))
```

Code 6.19: Util Module Functions

Conclusion

In this paper, we have expanded on the work presented in [20] regarding Physical Layer Security using Reconfigurable Intelligent Surfaces (RISs). We generalized the framework to support multiple receiving users and multiple RIS configurations, both in parallel and in series. By mathematically proving the formulas, and physically simulating realistic scenarios, we demonstrated the validity and usefulness of the prooposed work.

With our contribution, the framework is now able to manage:

- Multiple receivers in different positions
- Multiple RISs in parallel that increase signal quality and security
- Multiple RISs working in series to accomodate complex situation
- A wide combination of these properties in realistic network conditions

With our Bit Error Rate (BER) simulations, we proved and demostrated how the receivers are able to receive correctly the messages with a low error percenteage, while ensuring no other malicious actor can decypher the signal when not having direct Line of Sight (LOS) from the transmitter. Even when this link is present, our configurations ensure the RIS distrupt the interception of the signal with significant noise, even at high Signal to Noise Ratio (SNR).

We also showed the realistic application of our framework in a simulated scenario including realistic channel gain calculations, adding Rician fading and considering signal strenght using path loss. These added simulation will aid exporting our solution from a mathematical proof to an effectivle implementation usable for real life communication. We modeled different possibilities of path loss and RIS implementation to cover all possible variables, showing promising results even in the worst scenarios.

The implications of this work are particularly relevant for emerging technologies such as vehicular networks, Internet of Things, and other applications requiring secure wireless communications. Thanks to modern technologies, we are able to increase the security and privacy even at lower layers of communication, helping reducing the load on higher layers wich could impact negatively the usefulness of communications when latency and frequency of communication is crucial.

Future research directions could include:

- Further optimization of RIS configurations for dynamic environments with mobile nodes
- Integration with existing security protocols at higher network layers
- Usage of more complex communication protocol, like GSSK [12] instead of the proposed SSK [13]
- Implementation and testing in real-world scenarios, particularly in vehicular networks
- Extension to even more complex network topologies with multiple transmitters and heterogeneous receiver capabilities

In conclusion, our extended framework for physical layer security using RISs provides a promising approach to secure modern wireless communication systems, especially in scenarios where traditional encryption methods may introduce unacceptable computational overhead or latency. The flexibility to support multiple users and complex reflection paths makes it adaptable to various practical deployment scenarios while maintaining strong security guarantees.

Bibliography

[1] Yun Ai, Michael Cheffena, Aashish Mathur, and Hongjiang Lei. On physical layer security of double rayleigh fading channels for vehicular communications. *IEEE Wireless Communications Letters*, 7(6):1038–1041, Dec 2018.

[2] Ertugrul Basar, Marco Di Renzo, Julien De Rosny, Merouane Debbah, Mohamed-Slim Alouini, and Rui Zhang. Wireless communications through reconfigurable intelligent surfaces. *IEEE Access*, 7:116753–116773, 2019.

[3] Jie Chen, Ying-Chang Liang, Yiyang Pei, and Huayan Guo. Intelligent reflecting surface: A programmable wireless environment for physical layer security. *IEEE Access*, 7:82599–82612, 2019.

[4] Steven Dalton, Luke Olson, and Nathan Bell. Optimizing sparse matrix—matrix multiplication for the gpu. *ACM Trans. Math. Softw.*, 41(4), October 2015.

[5] Min Deng, Manzoor Ahmed, Abdul Wahid, Aized Amin Soofi, Wali Ullah Khan, Fang Xu, Muhammad Asif, and Zhu Han. Reconfigurable intelligent surfaces enabled vehicular communications: A comprehensive survey of recent advances and future challenges. *IEEE Transactions on Intelligent Vehicles*, pages 1–28, 2024.

[6] Mohamed A. ElMossallamy, Hongliang Zhang, Lingyang Song, Karim G. Seddik, Zhu Han, and Geoffrey Ye Li. Reconfigurable intelligent surfaces for wireless communications: Principles, challenges, and opportunities. *IEEE Transactions on Cognitive Communications and Networking*, 6(3):990–1002, Sep. 2020.

[7] Math Stack Exchange. How is the null space related to singular value decomposition? `https://math.stackexchange.com/questions/1771013/how-is-the-null-space-related-to-singular-value-decomposition`.

[8] S. Goel and R. Negi. Secret communication in presence of colluding eavesdroppers. In *MILCOM 2005 - 2005 IEEE Military Communications Conference*, pages 1501–1506 Vol. 3, Oct 2005.

[9] Satashu Goel and Rohit Negi. Guaranteeing secrecy using artificial noise. *IEEE Transactions on Wireless Communications*, 7(6):2180–2189, June 2008.

[10] Xiang He and Aylin Yener. Providing secrecy when the eavesdropper channel is arbitrarily varying: A case for multiple antennas. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1228–1235, Sep. 2010.

[11] Zhen-Qing He and Xiaojun Yuan. Cascaded channel estimation for large intelligent metasurface assisted massive mimo. *IEEE Wireless Communications Letters*, 9(2):210–214, Feb 2020.

[12] Jeyadeepan Jeganathan, Ali Ghrayeb, and Leszek Szczecinski. Generalized space shift keying modulation for mimo channels. In *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, Sep. 2008.

[13] Jeyadeepan Jeganathan, Ali Ghrayeb, Leszek Szczecinski, and Andres Ceron. Space shift keying modulation for mimo channels. *IEEE Transactions on Wireless Communications*, 8(7):3692–3703, July 2009.

[14] Jungi Jeong, Jun Hwa Oh, Seung Yoon Lee, Yuntae Park, and Sang-Hyuk Wi. An improved path-loss model for reconfigurable-intelligent-surface-aided wireless communications and experimental validation. *IEEE Access*, 10:98065–98078, 2022.

[15] Dimitrios S. Karas, Alexandros–Apostolos A. Boulogeorgos, and George K. Karagiannidis. Physical layer security with uncertainty on the location of the eavesdropper. *IEEE Wireless Communications Letters*, 5(5):540–543, Oct 2016.

[16] Ravneet Kaur, Bajrang Bansal, Sudhan Majhi, Sandesh Jain, Chongwen Huang, and Chau Yuen. A survey on reconfigurable intelligent surface for physical layer security of next-generation wireless communications. *IEEE Open Journal of Vehicular Technology*, 5:172–199, 2024.

[17] Steven Kisseleff, Wallace A. Martins, Hayder Al-Hraishawi, Symeon Chatzinotas, and Björn Ottersten. Reconfigurable intelligent surfaces for smart cities: Research challenges and opportunities. *IEEE Open Journal of the Communications Society*, 1:1781–1797, 2020.

[18] Sushil Kumar, Upasana Dohare, Kirshna Kumar, Durga Prasad Dora, Kashif Naseer Qureshi, and Rupak Kharel. Cybersecurity measures for geocasting in vehicular cyber physical system environments. *IEEE Internet of Things Journal*, 6(4):5916–5926, Aug 2019.

[19] Ruizhe Long, Ying-Chang Liang, Yiyang Pei, and Erik G. Larsson. Active reconfigurable intelligent surface-aided wireless communications. *IEEE Transactions on Wireless Communications*, 20(8):4962–4975, Aug 2021.

[20] Junshan Luo, Fanggang Wang, Shilian Wang, Hao Wang, and Dong Wang. Reconfigurable intelligent surface: Reflection design against passive eavesdropping. *IEEE Transactions on Wireless Communications*, 20(5):3350–3364, May 2021.

[21] Abubakar U. Makarfi, Khaled M. Rabie, Omprakash Kaiwartya, Kabita Adhikari, Xingwang Li, Marcela Quiroz-Castellanos, and Rupak Kharel. Reconfigurable intelligent surfaces-enabled vehicular networks: A physical layer security perspective, 2020.

[22] Amitav Mukherjee, S. Ali A. Fakoorian, Jing Huang, and A. Lee Swindlehurst. Principles of physical layer security in multiuser wireless networks: A survey. *IEEE Communications Surveys and Tutorials*, 16(3):1550–1573, Third 2014.

[23] Annapurna Pradhan, Susmita Das, Md Jalil Piran, and Zhu Han. A survey on physical layer security of ultra/hyper reliable low latency communication in 5g and 6g networks: Recent advancements, challenges, and future directions. *IEEE Access*, 12:112320–112353, 2024.

[24] Michele Segata, Paolo Casari, Marios Lestas, Alexandros Papadopoulos, Dimitrios Tyrovolas, Taqwa Saeed, George Karagiannidis, and Christos Liaskos. Cooperis: A framework for the simulation of reconfigurable intelligent surfaces in cooperative driving environments. *Computer Networks*, 248:110443, 2024.

[25] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, Oct 1949.

[26] Yi-Sheng Shiu, Shih Yu Chang, Hsiao-Chun Wu, Scott C.-H. Huang, and Hsiao-Hwa Chen. Physical layer security in wireless networks: a tutorial. *IEEE Wireless Communications*, 18(2):66–74, April 2011.

[27] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Nov 1994.

[28] Chris Sperandio and Paul G. Flikkema. Wireless physical-layer security via transmit precoding over dispersive channels: Optimum linear eavesdropping. pages 1113–1117, 2002. 2002 MILCOM Proceedings; Global Information GRID - Enabling Transformation Through 21st Century Communications ; Conference date: 07-10-2002 Through 10-10-2002.

[29] Ludwig Streloke and Jörg Franke. Electrifying the road: Disruptive shifts in automotive value creation. In *2024 1st International Conference on Production Technologies and Systems for E-Mobility (EPTS)*, pages 1–8, June 2024.

[30] Xiao Tang, Dawei Wang, Ruonan Zhang, Zheng Chu, and Zhu Han. Jamming mitigation via aerial reconfigurable intelligent surface: Passive beamforming and deployment optimization. *IEEE Transactions on Vehicular Technology*, 70(6):6232–6237, June 2021.

[31] Wade Trappe. The challenges facing physical layer security. *IEEE Communications Magazine*, 53(6):16–20, June 2015.

[32] David Tse and Pramod Viswanath. Fundamentals of wireless communication. `https://web.stanford.edu/~dntse/Chapters_PDF/Fundamentals_Wireless_Communication_chapter7.pdf`, 2005.

[33] Wikipedia. Free space path loss. `https://en.wikipedia.org/wiki/Free-space_path_loss`.

[34] Wikipedia. Rice distribution. `https://en.wikipedia.org/wiki/Rice_distribution`.

[35] Wikipedia. Rician fading. `https://en.wikipedia.org/wiki/Rician_fading`.

[36] Janghyuk Youn, Woong Son, and Bang Chul Jung. Physical-layer security improvement with reconfigurable intelligent surfaces for 6g wireless communication systems. *Sensors*, 21(4), 2021.