ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Giao thức phân phối khóa lượng tử BB84 dưới ảnh hưởng của nhiễu loạn khí quyển

LƯƠNG VĂN DƯƠNG
Duong.LV213854@sis.hust.edu.vn

Ngành Kỹ thuật điện tử-viễn thông

Giảng viên hướng dẫn: PGS. TS. Hà Duyên Trung

Chữ ký của GVHD

PGS. TS. Đỗ Trọng Tuấn

Chữ ký của GVHD

KHOA: Kỹ thuật truyền thông

HÀ NỘI, 7/2025

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN – ĐIỆN TỬ

ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP

(DÀNH CHO CÁN BỘ HƯỚNG DẪN)

Tên đề tài: Giao thức phân phối khóa lượng tử BB84 dưới ảnh hưởng của nhiễu loạn khí quyển.

Họ tên SV: LƯƠNG VĂN DƯƠNG **MSSV:** 20213854

Cán bộ hướng dẫn 1: PGS. TS. HÀ DUYÊN TRUNG

Cán bộ hướng dẫn 2: PGS. TS. ĐỖ TRỌNG TUẤN

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	**Thái độ làm việc (2,5 điểm)**	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN	
		Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	**Kỹ năng viết quyển ĐATN (2 điểm)**	Trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.	
		Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	**Nội dung và kết quả đạt được (5 điểm)**	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.	

		Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.	
		Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	**Điểm thành tích (1 điểm)**	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. **(1 điểm)**	
		Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. **(0,5 điểm)**	
	Điểm tổng các tiêu chí:		
	Điểm hướng dẫn:		

Cán bộ hướng dẫn
(Ký và ghi rõ họ tên)

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN – ĐIỆN TỬ

ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP

(DÀNH CHO CÁN BỘ PHẢN BIỆN)

Tên đề tài: Giao thức phân phối khóa lượng tử BB84 dưới ảnh hưởng của nhiễu loạn khí quyển.

Họ tên SV: Lương Văn Dương **MSSV:** 20213854

Cán bộ phản biện: ……..

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	**Trình bày quyển ĐATN (4 điểm)**	Đồ án trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.	
		Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
2	**Nội dung và kết quả đạt được (5,5 điểm)**	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.	
		Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả	

		trước đó có liên quan.	
		Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
3	**Điểm thành tích (1 điểm)**	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ các giải thưởng KH quốc tế/ trong nước từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. **(1 điểm)**	
		Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành. **(0,5 điểm)**	
	Điểm tổng các tiêu chí:		
	Điểm phản biện:		

Cán bộ phản biện

(Ký và ghi rõ họ tên)

Lời cảm ơn

Em xin gửi lời tri ân sâu sắc đến Ban Giám hiệu cùng toàn thể thầy cô Trường Điện-Điện tử, Đại học Bách khoa Hà Nội, đặc biệt là các thầy cô Khoa Điện tử và Khoa Kĩ thuật truyền thông, những người đã tận tâm giảng dạy và truyền đạt những kiến thức quý báu cho em trong suốt quá trình học tập.

Đặc biệt, em xin bày tỏ lòng biết ơn đến hai thầy hướng dẫn, Hà Duyên Trung và Đỗ Trọng Tuấn, những người đã luôn tận tình hỗ trợ, đồng hành và khích lệ em trong suốt quá trình thực hiện khóa luận tốt nghiệp. Bên cạnh đó, em cũng trân trọng cảm ơn gia đình và bạn bè, những người đã luôn bên cạnh, động viên và tiếp thêm sức mạnh để em có thể vượt qua thử thách và hoàn thành tốt nhiệm vụ học tập.

Em chân thành cảm ơn tất cả sự giúp đỡ và đóng góp quý báu của thầy cô!

Tóm tắt nội dung đồ án

Đồ án nghiên cứu và mô phỏng giao thức phân phối khóa lượng tử BB84 trong môi trường nhiễu fading Gamma-Gamma. Nội dung chính của đồ án gồm: phân tích lý thuyết về mật mã lượng tử, giao thức BB84 và ảnh hưởng của nhiễu fading Gamma-Gamma đến quá trình truyền khóa lượng tử. Từ đó, xây dựng mô hình toán học mô phỏng hệ thống BB84 trong môi trường truyền thông nhiễu thực tế. Hệ thống được mô phỏng nhằm đánh giá các chỉ số hiệu năng như tỷ lệ lỗi bit lượng tử (QBER), hiệu suất sinh khóa và tác động của kẻ nghe lén (Eve). Đồ án cũng xác định kích thước truyền khóa tối ưu, nhằm cân bằng giữa bảo mật và độ ổn định trong điều kiện fading. Ngoài ra, đồ án đề xuất một số hướng phát triển như: tối ưu hóa công suất nguồn phát, cải thiện kỹ thuật điều chỉnh kênh truyền, và tích hợp hệ thống vào các nền tảng truyền thông thế hệ mới như mạng 6G bảo mật lượng tử.

Kết quả cho thấy fading Gamma-Gamma gây ảnh hưởng đáng kể đến chất lượng phân phối khóa lượng tử, đòi hỏi các phương pháp điều chỉnh thích ứng để đảm bảo bảo mật. Đồng thời, mô phỏng khẳng định rằng chỉ số QBER có thể đóng vai trò chỉ thị hiệu quả trong việc phát hiện sự hiện diện của kẻ nghe lén, từ đó nâng cao mức độ an toàn cho các hệ thống truyền thông lượng tử thực tế.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

ABSTRACT

This thesis focuses on the study and simulation of the BB84 quantum key distribution (QKD) protocol in the presence of Gamma-Gamma fading. The main content includes a theoretical analysis of quantum cryptography, the BB84 protocol, and the impact of Gamma-Gamma fading on the quantum key distribution process. A mathematical model is developed to simulate the BB84 system under realistic noisy communication conditions.The simulation aims to evaluate key performance metrics such as the quantum bit error rate (QBER), key generation efficiency, and the impact of eavesdroppers (Eve). The thesis also identifies the optimal key transmission size to balance security and stability under fading conditions. Furthermore, several development directions are proposed, including optimization of source power, improvement of channel adaptation techniques, and integration of the system into next-generation communication platforms such as quantum-secure 6G networks.

The results show that Gamma-Gamma fading significantly affects the quality of quantum key distribution, highlighting the need for adaptive countermeasures to ensure secure communication. The simulation also confirms that QBER can serve as an effective indicator for detecting the presence of eavesdroppers, thereby enhancing the security level of practical quantum communication systems.

LỜI CAM KẾT

Em là Lương Văn Dương, mã số sinh viên 20213854, sinh viên lớp Điện tử 07, khóa 66, Trường Điện – Điện tử, Đại học Bách Khoa Hà Nội. Người hướng dẫn là PGS.TS. Hà Duyên Trung và PGS.TS. Đỗ Trọng Tuấn. Em xin cam đoan toàn bộ nội dung trình bày trong đồ án " **Giao thức phân phối khóa lượng tử BB84 dưới ảnh hưởng của nhiễu loạn khí quyển**" là kết quả quá trình tìm hiểu và nghiên cứu của em. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc và kiểm thử thực tế. Mọi thông tin được trích dẫn đều tuân thủ quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Em xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, ngày 27 tháng 6 năm 2025

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

THUẬT NGỮ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Giải thích tiếng Việt
QKD	Quantum Key Distribution	Phân phối khóa lượng tử – kỹ thuật chia sẻ khóa bảo mật dựa trên các nguyên lý lượng tử.
BB84	Bennett-Brassard 1984 Protocol	Giao thức QKD đầu tiên và phổ biến nhất, đề xuất bởi Bennett và Brassard năm 1984.
QBER	Quantum Bit Error Rate	Tỷ lệ lỗi bit lượng tử – chỉ số đánh giá độ tin cậy của quá trình truyền khóa lượng tử.
CV-QKD	Continuous Variable QKD	QKD sử dụng các biến liên tục như biên độ và pha để mã hóa thông tin.
DV-QKD	Discrete Variable QKD	QKD sử dụng các trạng thái rời rạc (như phân cực photon) để truyền khóa.
VLC	Visible Light Communication	Truyền thông ánh sáng khả kiến – công nghệ truyền dữ liệu bằng đèn LED.
FSO	Free Space Optics	Truyền thông quang học trong không gian tự do.
PDF	Probability Density Function	Hàm mật độ xác suất – dùng để mô tả phân bố của biến ngẫu nhiên liên tục.
LED	Light Emitting Diode	Đi-ốt phát quang – nguồn phát tín hiệu trong hệ thống VLC.
PD	Photodetector	Bộ thu quang – thiết bị nhận và chuyển đổi ánh sáng thành tín hiệu điện.
Eve	Eavesdropper	Kẻ nghe lén – thường được mô phỏng để kiểm tra khả năng bảo mật của giao thức.

Alice	(tên nhân vật)	Người gửi khóa lượng tử trong giao thức QKD.
Bob	(tên nhân vật)	Người nhận khóa lượng tử trong giao thức QKD.
BPSK	Binary Phase Shift Keying	Điều chế dịch pha nhị phân – kỹ thuật mã hóa dữ liệu bằng cách thay đổi pha sóng.
FOV	Field of View	Góc nhìn hiệu dụng – góc mà thiết bị thu có thể nhận tín hiệu.
LOS	Line of Sight	Truyền thẳng – tín hiệu truyền trực tiếp từ nguồn phát đến đích không bị chắn.
SNR	Signal-to-Noise Ratio	Tỷ số tín hiệu trên nhiễu – chỉ số đánh giá độ mạnh tín hiệu so với nhiễu nền.
IT	Photodetector Current	Dòng điện tại photodetector – tín hiệu điện sinh ra khi ánh sáng tới bộ thu.
MATLAB	Matrix Laboratory	Phần mềm dùng để mô phỏng, xử lý tín hiệu và lập trình tính toán kỹ thuật.
Streamlit	Streamlit Framework	Thư viện Python dùng để xây dựng giao diện web trực quan cho các ứng dụng mô phỏng.

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

Chương này trình bày bối cảnh, lý do lựa chọn đề tài, mục tiêu nghiên cứu, định hướng phương pháp và các kết quả chính đã đạt được. Đây là nền tảng để người đọc hiểu được tính cấp thiết và hướng tiếp cận của đồ án.

1.1 Đặt vấn đề

Trong bối cảnh kỷ nguyên số phát triển mạnh mẽ, vấn đề bảo mật thông tin đang trở nên ngày càng cấp thiết và mang tính sống còn đối với mọi lĩnh vực – từ chính phủ, tài chính, quốc phòng, đến hạ tầng dân sự và các hệ thống IoT hiện đại. Các hệ mật mã cổ điển như RSA, AES hay ECC đã và đang được sử dụng phổ biến trong hầu hết các hệ thống truyền thông hiện nay. Tuy nhiên, với sự ra đời và phát triển nhanh chóng của các máy tính lượng tử – vốn có khả năng xử lý song song vượt trội – các thuật toán cổ điển vốn an toàn trong hàng chục năm qua đang đứng trước nguy cơ bị phá vỡ. Thuật toán Shor, với khả năng phân tích số nguyên lớn, có thể đánh bại RSA và ECC trong thời gian ngắn; trong khi thuật toán Grover có thể làm suy yếu đáng kể độ an toàn của các hệ mã hóa đối xứng như AES. Trước bối cảnh đó, nhu cầu tìm kiếm các giải pháp bảo mật mới, có khả năng chống lại các tấn công lượng tử, trở nên cấp bách hơn bao giờ hết.

Một trong những hướng tiếp cận mang tính cách mạng là phân phối khóa lượng tử (Quantum Key Distribution – QKD). Không giống các hệ mật mã dựa trên giả thuyết tính toán, QKD tận dụng các nguyên lý cơ bản của cơ học lượng tử, chẳng hạn như nguyên lý bất định Heisenberg và định lý không nhân bản (no-cloning theorem), để đảm bảo việc chia sẻ khóa giữa hai bên là tuyệt đối an toàn về mặt lý thuyết. Cụ thể, trong hệ thống QKD, bất kỳ sự xâm nhập nào từ kẻ tấn công thứ ba (Eve) cũng sẽ làm thay đổi trạng thái lượng tử của hạt truyền, từ đó gây ra lỗi có thể phát hiện được trong quá trình kiểm tra khóa. Điều này giúp hai bên (Alice và Bob) không chỉ phát hiện được sự hiện diện của kẻ nghe lén mà còn chủ động loại bỏ các bit bị rò rỉ trước khi sử dụng khóa để mã hóa thông tin thực.

Trong số các giao thức QKD đã được phát triển, BB84 – do Charles Bennett và Gilles Brassard đề xuất vào năm 1984 – là giao thức đầu tiên và đến nay vẫn là một trong những giao thức phổ biến và dễ triển khai nhất. BB84 mã hóa các bit nhị phân thành các trạng thái lượng tử (ví dụ: phân cực photon theo các cơ sở khác nhau) và sử dụng phép đo ngẫu nhiên để kiểm tra tính toàn vẹn của kênh. Chỉ số lỗi bit lượng tử (Quantum Bit Error Rate – QBER) được sử dụng như một tiêu chí để đánh giá mức độ an toàn của quá trình phân phối khóa: nếu QBER vượt quá một ngưỡng nhất định, khóa sẽ bị loại bỏ hoàn toàn; ngược lại, nếu trong giới hạn cho phép, khóa sẽ được xử lý thêm bằng các thuật toán sửa lỗi và trích xuất bảo mật để đảm bảo tính toàn vẹn.

Tuy nhiên, để đảm bảo tính khả thi và hiệu quả trong thực tế, các hệ thống QKD như BB84 cần được triển khai trên những kênh truyền cụ thể – và chính đặc điểm vật lý của những kênh này sẽ ảnh hưởng sâu sắc đến hiệu suất và độ an toàn của hệ thống. Trong môi trường truyền thông quang học không dây (ví dụ: liên kết giữa vệ tinh, UAV hoặc các thiết bị mặt đất), tín hiệu ánh sáng thường xuyên phải đối mặt với các hiện tượng gây nhiễu như: nhiễu xạ, dao động áp suất, nhiệt

độ và các chuyển động môi trường. Những yếu tố này gây ra hiện tượng fading – tức sự biến thiên ngẫu nhiên theo thời gian và không gian của cường độ tín hiệu nhận được. Trong đó, mô hình fading Gamma-Gamma là một mô hình thống kê được sử dụng phổ biến để mô tả chính xác hiệu ứng nhiễu loạn trong các kênh quang học, đặc biệt trong điều kiện nhiễu trung bình đến mạnh. Mô hình này kết hợp cả hai hiệu ứng nhỏ (do nhiễu xạ cục bộ) và lớn (do biến thiên quỹ đạo truyền), làm cho hệ thống trở nên khó dự đoán và dễ sai lệch nếu không có điều chỉnh phù hợp.

Chính vì vậy, việc nghiên cứu và mô phỏng giao thức BB84 trong môi trường fading Gamma-Gamma không chỉ có ý nghĩa lý thuyết mà còn đóng vai trò quan trọng trong việc đưa các giải pháp bảo mật lượng tử vào ứng dụng thực tế, mở ra triển vọng về một hạ tầng an toàn trước các nguy cơ tấn công lượng tử trong tương lai.

1.2 Mục tiêu đề ra

Mục tiêu tổng quát của đề tài là phát triển một mô hình mô phỏng hoàn chỉnh nhằm đánh giá hiệu suất và tính khả thi của giao thức phân phối khóa lượng tử BB84 trong điều kiện kênh truyền có nhiễu fading mô phỏng theo phân bố Gamma-Gamma. Việc kết hợp giao thức lượng tử với mô hình kênh thực tế hướng đến việc hiểu rõ hơn tác động của môi trường truyền không lý tưởng đến quá trình trao đổi khóa lượng tử và đề xuất các giải pháp kỹ thuật phù hợp để duy trì mức độ bảo mật mong muốn. Để hiện thực hóa mục tiêu đó, đề tài đặt ra các mục tiêu cụ thể như sau:

- *Mục tiêu 1:* Hệ thống hóa các kiến thức lý thuyết cần thiết về phân phối khóa lượng tử, đặc biệt tập trung vào giao thức BB84, bao gồm quá trình phát, đo, sàng lọc khóa, đánh giá sai số và xử lý bảo mật cuối cùng. Các bước từ tạo bit lượng tử đến sinh khóa cuối sẽ được trình bày rõ ràng theo trình tự và logic toán học chặt chẽ.

- *Mục tiêu 2:* Phân tích và áp dụng mô hình fading Gamma-Gamma vào mô tả kênh truyền tín hiệu lượng tử. Đây là một mô hình xác suất hai tham số có khả năng phản ánh tốt các hiệu ứng nhiễu xạ và dao động biên độ tín hiệu, đặc biệt trong các kênh truyền quang không dây hoặc môi trường biến động cường độ trung bình đến mạnh. Việc thay đổi tham số mô hình nhằm tái hiện nhiều điều kiện môi trường khác nhau.

- *Mục tiêu 3:* Phát triển chương trình mô phỏng số để mô hình hóa hệ thống BB84 với kênh truyền chịu ảnh hưởng của fading Gamma-Gamma. Hệ thống sẽ được triển khai trong không gian giả lập, nơi các photon được truyền từ nguồn phát đến thiết bị thu trong điều kiện nhiễu thống kê. Từ đó, các chỉ số quan trọng như QBER, tỷ lệ khóa sinh cuối cùng, và xác suất rò rỉ khóa sẽ được tính toán và phân tích định lượng.

- *Mục tiêu 4:* Thực hiện các thí nghiệm mô phỏng trên nhiều cấu hình kênh và mức độ fading khác nhau, từ đó rút ra được mối tương quan giữa cường độ fading và hiệu suất của giao thức BB84. Điều này giúp xác định các ngưỡng vận hành an toàn, cũng như chỉ ra các điều kiện môi trường có thể gây rủi ro

bảo mật cao.

- *Mục tiêu 5:* Đề xuất các chiến lược kỹ thuật giúp giảm thiểu tác động bất lợi từ môi trường truyền, chẳng hạn như điều chỉnh tần suất truyền khóa, cấu hình lại tỷ lệ lựa chọn cơ sở đo, hoặc bổ sung quy trình kiểm tra ngưỡng QBER động để chủ động phát hiện sự cố trong mạng lượng tử.

1.3 Định hướng phương pháp nghiên cứu

Định hướng chính của đề tài là tiếp cận vấn đề từ góc nhìn kết hợp giữa lý thuyết bảo mật lượng tử và mô hình hóa kênh truyền trong thực tế nhằm đưa ra cái nhìn toàn diện về hiệu suất vận hành của giao thức BB84 khi triển khai trong điều kiện môi trường không lý tưởng. Đề tài không chỉ hướng tới việc mô phỏng, mà còn đặt trọng tâm vào phân tích và diễn giải các chỉ số ảnh hưởng đến độ an toàn và hiệu quả phân phối khóa. Phương pháp nghiên cứu sẽ được triển khai qua các giai đoạn cụ thể như sau:

- *Giai đoạn 1* – Nghiên cứu cơ sở lý thuyết: Thu thập, phân tích và hệ thống hóa các tài liệu khoa học liên quan đến giao thức BB84, các nguyên lý vật lý lượng tử cơ bản (như nguyên lý bất định, không nhân bản), các bước xử lý khóa (sàng lọc, sửa lỗi, khuếch đại độ riêng tư), cũng như phương pháp đánh giá độ bảo mật bằng QBER và entropy khóa. Việc hiểu vững các nền tảng này là tiền đề quan trọng để mô hình hóa chính xác hệ thống lượng tử trong môi trường có nhiễu.

- *Giai đoạn 2* – Mô hình hóa kênh truyền fading Gamma-Gamma: Xây dựng mô hình toán học mô tả xác suất phân bố biên độ tín hiệu dưới tác động của fading Gamma-Gamma. Mô hình sẽ bao gồm hai tham số α và β đại diện cho mức độ nhiễu xạ và dao động cường độ, đồng thời tích hợp vào quá trình truyền photon trong hệ thống BB84. Ngoài ra, các giả định về nguồn phát, máy thu, thiết bị đo và tỷ lệ chọn cơ sở sẽ được thiết lập phù hợp với các mô hình thực nghiệm đã công bố trong các công trình khoa học trước đó.

- *Giai đoạn 3* – Thiết kế mô phỏng số: Sử dụng phần mềm MATLAB hoặc Python để xây dựng chương trình mô phỏng toàn bộ quá trình phân phối khóa BB84 trên kênh Gamma-Gamma. Mô phỏng sẽ được tổ chức theo nhiều phiên truyền khóa, mỗi phiên gồm hàng nghìn bit lượng tử để đảm bảo tính thống kê. Các thông số như xác suất lỗi đo, tỷ lệ trùng khớp cơ sở, và các tham số môi trường sẽ được tinh chỉnh linh hoạt nhằm quan sát toàn diện các kịch bản khác nhau.

- *Giai đoạn 4* – Phân tích kết quả và đánh giá hiệu suất: Từ dữ liệu mô phỏng, các chỉ số chính như QBER, tỷ lệ sinh khóa cuối cùng, và độ tin cậy của hệ thống sẽ được rút trích và phân tích. Đặc biệt, biểu đồ QBER theo các mức fading sẽ được xây dựng nhằm tìm ra mối tương quan giữa môi trường và khả năng bảo mật. Ngoài ra, vai trò của QBER như một công cụ phát hiện sự xâm nhập từ Eve cũng sẽ được kiểm tra thông qua mô hình hóa các tình huống nghe lén có chủ đích.

- *Giai đoạn 5* – Đề xuất hướng cải tiến và ứng dụng: Dựa trên kết quả mô

phỏng, đề tài sẽ đưa ra các khuyến nghị kỹ thuật giúp nâng cao hiệu suất và độ tin cậy của giao thức BB84 trong môi trường fading. Một số hướng có thể bao gồm: thay đổi chiến lược lựa chọn bit kiểm tra lỗi, xây dựng hệ thống cảnh báo khi QBER vượt ngưỡng, và nghiên cứu các giao thức QKD khác có khả năng kháng nhiễu tốt hơn trong tương lai.

1.4 Kết quả đạt được

1.4.1 Các giả định và giới hạn của đề tài

Trong quá trình xây dựng mô hình và thực hiện mô phỏng giao thức phân phối khóa lượng tử BB84 dưới ảnh hưởng của fading Gamma-Gamma, đề tài đã đưa ra một số giả định nhằm đơn giản hóa vấn đề và đảm bảo tính khả thi trong phạm vi thời gian và công cụ thực hiện. Tuy nhiên, những giả định này cũng đồng thời tạo ra một số giới hạn nhất định về tính ứng dụng và độ chính xác tuyệt đối của kết quả mô phỏng. Các giả định và giới hạn của đề tài như sau:

Môi trường trường truyền đơn tuyến: Đề tài giả định quá trình truyền tín hiệu lượng tử diễn ra trong một kênh quang học đơn tuyến giữa hai thực thể là Alice và Bob, không xét đến các yếu tố như phản xạ đa đường, tán xạ không gian, hay mất mát do định hướng sai lệch.

Không xét đến lỗi thiết bị vật lý: Các thiết bị tại Alice, Bob và Eve được giả định là hoạt động lý tưởng, không có sai số do hiệu chuẩn, không có trễ thời gian, và khả năng phát hiện hoặc phát xạ qubit là chính xác tuyệt đối (trừ các yếu tố nhiễu được mô hình hóa bởi fading Gamma-Gamma).

Fading ảnh hưởng tại đầu thu (Bob): Mô hình fading Gamma-Gamma chỉ được áp dụng tại phía người nhận (Bob), ảnh hưởng đến quá trình đo photon nhận được. Tác động của fading lên quá trình phát (Alice) hoặc truyền dẫn giữa chặng trung gian (nếu có) không được xét đến.

Kẻ nghe lén (Eve) thực hiện tấn công kiểu intercept-resend: Trong mô phỏng có kẻ nghe lén, Eve được giả định sử dụng chiến thuật đo và gửi lại (intercept-resend) với xác suất cơ sở ngẫu nhiên. Các hình thức tấn công nâng cao hơn như tấn công đo chọn lọc (measurement-device-independent), tấn công theo thời gian (timing attack) hoặc giả mạo thiết bị không được xem xét.

Tỷ lệ chọn cơ sở đo là ngẫu nhiên đều: Cả Alice và Bob đều lựa chọn cơ sở đo (Z hoặc X) với xác suất 50%. Điều này phù hợp với định nghĩa gốc của giao thức BB84 và giúp đảm bảo tính đối xứng trong xử lý số liệu.

Không xét các bước xử lý khóa nâng cao: Đề tài chưa thực hiện mô phỏng chi tiết các bước hậu xử lý như sửa lỗi (error correction) hoặc khuếch đại độ riêng tư (privacy amplification) – vốn là phần quan trọng để tạo ra khóa an toàn cuối cùng. Việc này có thể được mở rộng trong các nghiên cứu tiếp theo.

Chưa triển khai mô hình thực nghiệm vật lý: Mô phỏng được thực hiện hoàn toàn trong môi trường số, sử dụng các công cụ như Python và Streamlit. Các yếu tố vật lý như nhiễu nền thực tế, sai lệch chùm tia, mất mát quang học, hay chất lượng thiết bị đo thực tế chưa được tích hợp.

Mô hình fading còn lý tưởng hóa: Mặc dù fading Gamma-Gamma là một mô hình mạnh trong mô tả kênh quang, nhưng nó vẫn là mô hình thống kê lý tưởng. Việc áp dụng mô hình này vào kênh lượng tử giả định rằng các hiệu ứng phi tuyến, tán xạ hẹp/góc rộng, và các yếu tố vật lý phức tạp khác không ảnh hưởng đến bản chất trạng thái lượng tử – đây là một giả định có thể gây sai số nếu áp dụng vào thực tế mà không hiệu chỉnh.

Chưa xét kịch bản truyền nhiều qubit đồng thời (MIMO, đa photon): Hệ thống mô phỏng hiện tại hoạt động trên từng qubit đơn lẻ, theo đúng định nghĩa cổ điển của BB84. Các hệ thống truyền lượng tử song song hoặc đa photon (multi-photon states) chưa được xem xét.

Số lượng bit mô phỏng còn giới hạn: Do giới hạn về thời gian xử lý và năng lực máy tính, các mô phỏng chủ yếu tập trung trong khoảng 100–1000 bit. Điều này có thể chưa đủ để đánh giá hiệu suất của hệ thống ở quy mô triển khai thực tế, vốn cần hàng triệu qubit cho mỗi phiên truyền khóa.

1.4.2 Kết quả đạt được

Trong quá trình thực hiện đồ án, nhiều kết quả quan trọng đã được đạt được, góp phần kiểm chứng hiệu suất của giao thức phân phối khóa lượng tử BB84 trong điều kiện kênh truyền chịu ảnh hưởng của mô hình fading Gamma-Gamma. Các kết quả bao gồm cả mô phỏng trong môi trường lý tưởng lẫn môi trường thực tế có yếu tố gây nhiễu và có sự can thiệp của kẻ nghe lén.

Đầu tiên, đồ án đã xây dựng thành công hệ thống mô phỏng trực quan giao thức BB84 sử dụng công cụ Python và Streamlit. Giao diện cho phép người dùng dễ dàng theo dõi từng bước trong quy trình truyền khóa lượng tử, từ khởi tạo bit ngẫu nhiên, chọn cơ sở đo, đo tín hiệu, đến lọc bit trùng khớp. Mô phỏng này giúp người học hiểu rõ hơn về nguyên lý hoạt động của BB84 và cũng là một công cụ hỗ trợ giảng dạy hiệu quả trong lĩnh vực mật mã lượng tử.

Tiếp theo, trong điều kiện mô phỏng lý tưởng (không có nhiễu, không có Eve), hệ thống cho thấy hiệu suất hoạt động rất tốt. Sau khi thực hiện truyền và sàng lọc khóa, Alice và Bob thu được chuỗi khóa hoàn toàn giống nhau, với tỷ lệ lỗi lượng tử (QBER) bằng 0% và hiệu suất sinh khóa đạt khoảng 43%. Điều này chứng minh rằng hệ thống mô phỏng tái hiện chính xác nguyên lý và logic hoạt động của BB84 trong điều kiện lý tưởng.

Đồ án cũng đã mở rộng mô hình bằng cách thêm vào mô phỏng trường hợp có kẻ nghe lén Eve. Khi Eve thực hiện tấn công kiểu "intercept-resend", QBER trong hệ thống tăng đáng kể (khoảng 14.81%), cho thấy khả năng phát hiện sự xâm nhập của giao thức BB84 là hoàn toàn khả thi. Việc so sánh khóa và các bit kiểm tra giữa Alice và Bob sau khi có sự can thiệp của Eve cho thấy chuỗi khóa không còn trùng khớp, qua đó có thể quyết định hủy khóa để đảm bảo bảo mật.

Một phần cốt lõi của đồ án là mô phỏng giao thức BB84 trong môi trường chịu ảnh hưởng của fading Gamma-Gamma. Mô hình này được xây dựng để mô tả hiện tượng suy giảm và dao động cường độ tín hiệu do nhiễu loạn môi trường, rất phù hợp với thực tiễn của các hệ thống truyền thông lượng tử trong không gian mở. Hệ số fading được sinh ngẫu nhiên từ phân phối Gamma-Gamma và ảnh

hưởng trực tiếp đến xác suất đo đúng của Bob. Kết quả cho thấy QBER có xu hướng giảm dần và ổn định khi kích thước truyền tăng, trong khi hiệu suất giữ khóa dao động quanh mức 50%, phù hợp với lý thuyết.

Thông qua quá trình thử nghiệm, đồ án đã xác định được kích thước truyền tối ưu (SIZE_TX = 400 bit) giúp cân bằng giữa độ chính xác (QBER thấp), hiệu suất giữ khóa cao và tỷ lệ giữ lại khóa tốt nhất. Đây là thông số có ý nghĩa thực tiễn quan trọng nếu muốn triển khai hệ thống BB84 trong môi trường thực tế.

Ngoài ra, mô phỏng với các cặp tham số (α, β) khác nhau cho thấy rằng khi α và β tăng – tức fading yếu dần – thì QBER giảm rõ rệt, phản ánh chính xác đặc tính vật lý của mô hình Gamma-Gamma. Điều này cung cấp cơ sở định lượng để đánh giá chất lượng môi trường truyền và tác động của các yếu tố khí quyển đến độ tin cậy của quá trình phân phối khóa.

Cuối cùng, mô phỏng BB84 trong điều kiện có cả fading Gamma-Gamma và sự hiện diện của Eve đã cho thấy hiệu ứng cộng hưởng bất lợi: QBER tăng lên khoảng 25–29%, cao hơn đáng kể so với khi chỉ có một yếu tố ảnh hưởng. Điều này khẳng định rằng QBER là một chỉ báo hiệu quả để phát hiện nghe lén ngay cả trong môi trường truyền kém lý tưởng, từ đó củng cố vị thế của BB84 như một giao thức có khả năng bảo mật mạnh trong các điều kiện thực tế.

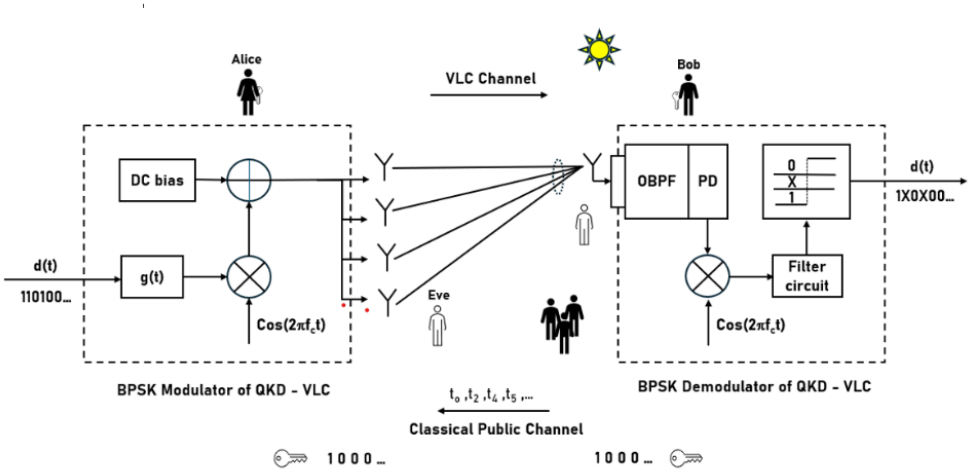
1.5 Kết luận chương

Chương 1 đã trình bày rõ bối cảnh thực tế, lý do nghiên cứu và các mục tiêu cụ thể của đề tài. Bên cạnh đó, định hướng nghiên cứu và các giả định ban đầu cũng đã được nêu rõ, tạo tiền đề cho các chương tiếp theo.

CHƯƠNG 2. TỔNG QUAN LÝ THUYẾT

Chương này cung cấp nền tảng lý thuyết cần thiết phục vụ cho việc mô phỏng hệ thống BB84, bao gồm kiến thức về QKD, giao thức BB84, các mô hình kênh truyền và phân phối xác suất fading Gamma-Gamma.

2.1 Phân phối khóa lượng tử biến liên tục CV-QKD



Hình 1- Sơ đồ tín hiệu

Hình 1 minh họa sơ đồ khối của hệ thống Continuous Variable Quantum Key Distribution over Visible Light Communication (CV/QKD-VLC), trong đó tích hợp công nghệ phân phối khóa lượng tử biến liên tục (CV-QKD) vào hệ thống truyền thông ánh sáng khả kiến (VLC) nhằm đảm bảo bảo mật thông tin trong quá trình truyền dữ liệu. Hệ thống bao gồm bốn thành phần chính: Alice (bên gửi), kênh truyền VLC, Bob (bên nhận) và kênh công khai để trao đổi thông tin cổ điển. Quá trình truyền thông bắt đầu tại Alice, nơi khóa lượng tử được tạo ra và ánh sáng LED được sử dụng để mã hóa thông tin. Cụ thể, Alice chuẩn bị các trạng thái quang lượng tử thông qua khối Quantum State Preparation, sử dụng bộ Optical Modulator để điều chế biên độ hoặc pha ánh sáng theo chuẩn CV-QKD, và sau đó phát tín hiệu thông qua LED Transmitter.

Tín hiệu này di chuyển qua kênh truyền VLC, nơi nó có thể bị ảnh hưởng bởi các yếu tố như suy hao đường truyền (path loss), nhiễu nền (background noise) từ môi trường xung quanh và tán xạ ánh sáng (scattering effects). Đáng chú ý, trong quá trình truyền dữ liệu, kẻ nghe lén Eve có thể cố gắng thu thập thông tin từ kênh VLC nhằm đánh cắp khóa lượng tử.Tại đầu nhận, Bob sử dụng một Photodetector (PD) để chuyển đổi tín hiệu ánh sáng thành dòng điện, sau đó áp dụng phương pháp đo lượng tử Homodyne Detection để trích xuất dữ liệu. Để đảm bảo khóa lượng tử được trích xuất chính xác, Bob thực hiện các bước xử lý dữ liệu cổ điển thông qua khối Classical Post-Processing, bao gồm lọc bit (Sifting), sửa lỗi (Error Correction) để đồng bộ hóa khóa, và khuếch đại bảo mật (Privacy Amplification) nhằm loại bỏ bất kỳ thông tin nào mà Eve có thể thu được.Ngoài kênh VLC, Alice và Bob còn sử dụng một kênh công khai để trao đổi thông tin về quá trình xử lý khóa. Mặc dù Eve có thể nghe lén kênh công khai này, nguyên lý cơ học lượng tử đảm bảo rằng bất kỳ sự can thiệp nào của Eve đều có thể được

phát hiện, nhờ đó hệ thống vẫn duy trì được tính bảo mật.

2.1.1 Tín hiệu điều chế trên sóng mang phụ

$$s(t) = a(t)r(t)\cos(2\pi f_c t) + \pi s_i \quad (1.1)$$

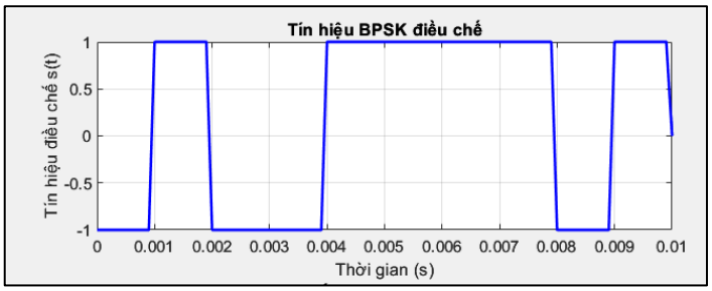
Công thức này mô tả tín hiệu điều chế dịch pha nhị phân (BPSK – *Binary Phase Shift Keying*) được truyền qua sóng mang phụ.

Trong đó:

- *s(t):* tín hiệu điều chế đầu ra.
- *a(t):* biên độ tín hiệu, quyết định mức năng lượng truyền đi.
- *r(t):* hàm xung chữ nhật, xác định khoảng thời gian bật/tắt của tín hiệu.
- f_c*:* tần số sóng mang (Hz), quyết định kênh truyền của tín hiệu.
- s_i*:* pha của tín hiệu, thay đổi theo bit dữ liệu (0 hoặc 1).

Ý nghĩa:

- Khi bit = 1, pha của tín hiệu bị dịch một góc π.
- Khi bit = 0, pha không thay đổi.
- Điều này giúp mã hóa dữ liệu số thành tín hiệu quang có thể truyền qua LED.



Hình 2- Tín hiệu điều chế trên sóng mang phụ

Hình 2 biểu diễn tín hiệu BPSK *s(t)* được tạo ra từ chuỗi bit nhị phân. Với mỗi bit 0 hoặc 1, pha của sóng mang được điều chỉnh lần lượt là 0 hoặc π, tạo ra sự đảo dấu giữa các chu kỳ sóng. Đồ thị thể hiện rõ sự thay đổi pha theo từng bit, tương ứng với dạng sóng cosine bị điều chế pha.

2.1.2 Công suất phát ra từ LED

$$P(t) = P[1 + \rho s(t)] \quad (1.2)$$

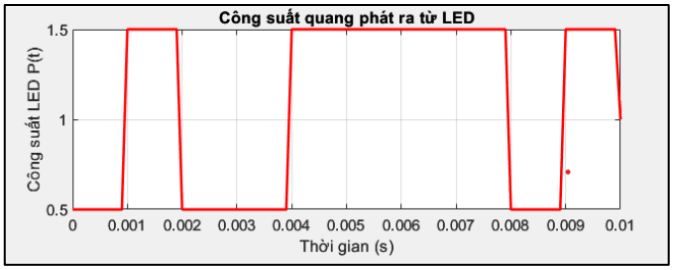
Trong đó:

- *P(t):* công suất LED tại thời điểm t.
- *P:* công suất phát cực đại của LED.

- *ρ:* độ sâu điều chế (modulation depth), xác định biên độ thay đổi của công suất.
- *s(t):* tín hiệu điều chế từ công thức (1).

Ý nghĩa:

- LED phát sáng theo tín hiệu điều chế.
- Nếu ρ quá lớn, tín hiệu có thể bị méo và gây lỗi.
- Nếu ρ quá nhỏ, độ tương phản thấp, làm giảm hiệu suất truyền dữ liệu.



Hình 3 - Mô phỏng công suất phát ra từ LED

Hình 3 biểu diễn công suất ánh sáng phát ra từ LED dưới tác động của tín hiệu điều chế *s(t)*. Công suất này không cố định mà dao động quanh giá trị trung bình, với mức dao động phụ thuộc vào hệ số điều chế ρ. Khi tín hiệu *s(t)* thay đổi do dữ liệu nhị phân đầu vào, công suất ánh sáng phát ra cũng thay đổi tương ứng. Dạng sóng trong hình cho thấy rõ mối liên hệ giữa tín hiệu số và công suất quang đầu ra, phản ánh cơ chế điều chế biên độ được sử dụng trong truyền thông bằng ánh sáng. Đây là cách để LED truyền tải thông tin bằng cách thay đổi cường độ phát sáng theo tín hiệu mang dữ liệu.

2.1.3 Độ lợi quang học của bộ thu

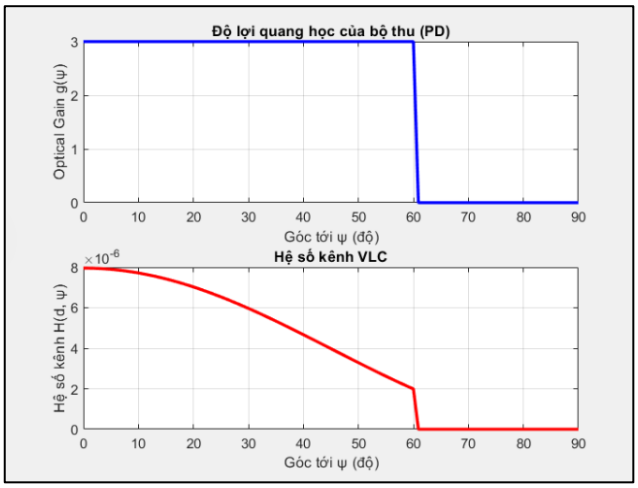
$$g(\psi) = \begin{cases} \dfrac{n^2}{\sin^2 \Psi_c}, & 0 \le \psi \le \Psi_c \\ 0, & \psi > \Psi_c \end{cases} \qquad (1.3)$$

Trong đó:

- $g(\psi)$: độ lợi quang học của photodetector (PD).
- n: chỉ số khúc xạ của thấu kính tại PD.
- Ψ_c: góc thu nhận tối đa của PD.
- ψ: góc tới của ánh sáng.

Ý nghĩa:

- Nếu $\psi \le \Psi_c$, PD thu được ánh sáng với độ lợi quang học nhất định.
- Nếu $\psi > \Psi_c$, tín hiệu bị triệt tiêu ($g(\psi) = 0$).



Hình 4 - Mô phỏng độ lợi quang học của bộ thu

Hình 4 mô tả đồ thị biểu diễn độ lợi quang học thu được tại photodiode (PD) theo góc tới ψ. Khi góc tới tăng, độ lợi duy trì ổn định trong vùng chấp nhận (FOV) và giảm xuống 0 hoàn toàn khi vượt quá góc giới hạn Ψ_c (trong ví dụ này là 60°). Điều này phản ánh khả năng thu ánh sáng của PD bị giới hạn bởi góc nhìn. Đồ thị còn cho thấy sự suy giảm của hệ số kênh theo góc tới ψ\psiψ. Khi góc càng lớn, tín hiệu đến PD bị yếu dần do cả hiệu ứng định hướng và giảm công suất do phân bố Lambertian của nguồn LED. Khi ψ vượt quá FOV, hệ số kênh giảm về 0, nghĩa là tín hiệu không đến được PD. *Hình 4* minh họa rõ mối quan hệ giữa hướng thu và chất lượng kênh truyền quang trong hệ thống VLC.

2.1.4 Hệ số truyền kênh VLC

$$H(d,\psi) = \begin{cases} \dfrac{(m+1)A}{2\pi d^2} cos^m(\phi)\, cos(\psi), & 0 \le \psi \le \Psi_c \\ 0, & \psi > \Psi_c \end{cases} \qquad (1.4)$$

Trong đó:

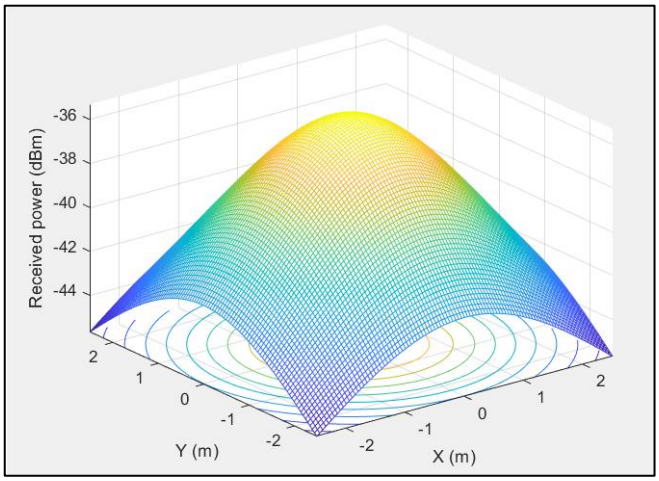
- $H(d,\psi)$: hệ số truyền kênh.
- d: khoảng cách giữa LED và photodetector PD.
- A: diện tích của PD.
- ψ: góc tới của ánh sáng.
- ϕ: góc phát của ánh sáng từ LED.
- m: chỉ số phân phối Lambertian được tính theo công thức:

$$m = \frac{\ln 2}{\ln \cos \phi_{1/2}} \qquad (1.5)$$

Ý nghĩa:

- Mô tả cách ánh sáng truyền từ LED đến bộ thu PD.

- Nếu khoảng cách quá xa, tín hiệu bị suy giảm nhanh theo d^2.
- Nếu góc tới quá lớn, PD không thu được ánh sáng.



Hình 5 - Mô phỏng hàm truyền kênh LOS

Hình 5 biểu diễn phân bố công suất quang thu được trên mặt phẳng thu trong một không gian phòng ba chiều, với nguồn sáng LED được đặt tại trung tâm trần phòng. Trục hoành (X) và tung (Y) thể hiện các vị trí trên mặt phẳng nhận, trong khi trục đứng (Z) biểu diễn công suất thu tại mỗi điểm theo đơn vị dBm. Đỉnh công suất xuất hiện ngay phía dưới nguồn LED và giảm dần khi di chuyển ra xa tâm, cho thấy ảnh hưởng rõ rệt của khoảng cách và góc tới đến chất lượng tín hiệu thu. Hình dạng đồ thị thể hiện sự suy hao công suất theo cả luật nghịch đảo bình phương và hiệu ứng định hướng (phụ thuộc vào $cos(\phi)$). Các đường màu trên mặt phẳng đáy (contour) hỗ trợ hình dung trực quan các vùng mạnh – yếu của tín hiệu quang. Qua đó, hình vẽ minh họa rõ ràng tính chất không đồng đều của kênh truyền trong hệ thống VLC và nhấn mạnh tầm quan trọng của việc bố trí hợp lý nguồn phát và thiết bị thu để tối ưu hiệu suất thu nhận tín hiệu.

2.1.5 Dòng điện nhận được tại PD

$$I(t) = \frac{\Re}{4} g(\psi) T_s(\psi) H(d,\psi) P_s(t) + n(t) \tag{1.6}$$

Trong đó:

- $I(t)$: dòng điện quang tại photodetector (PD).
- $\Re$: độ nhạy của PD (A/W).
- $g(\psi)$: độ lợi quang học của bộ thu.
- $T_s(\psi)$: hệ số lọc quang học.
- $H(d,\psi)$: hệ số truyền kênh VLC.
- $P_s(t)$: công suất tín hiệu nhận được tại PD.
- $n(t)$: nhiễu trong hệ thống (shot noise, thermal noise, background noise).

2.1.6 *Tín hiệu sau khi giải điều chế BPSK*

$$I_i(t) = \begin{cases} -\frac{\Re}{4} g(\psi) T_s(\psi) H(d,\psi) P_s(t) \rho + n(t), & \text{nếu } bit = 1 \\ \frac{\Re}{4} g(\psi) T_s(\psi) H(d,\psi) P_s(t) \rho + n(t), & \text{nếu } bit = 0 \end{cases} \quad (1.7)$$

Trong đó:

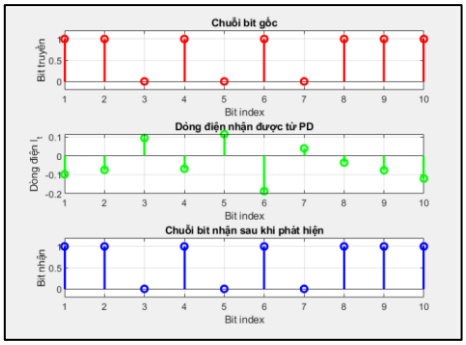
- Khi Alice gửi bit "1", tín hiệu có pha âm.
- Khi Alice gửi bit "0", tín hiệu có pha dương.
- Thành phần nhiễu $n(t)$ có thể làm sai lệch quyết định của Bob.

2.1.7 *Quy tắc phát hiện bit*

$$X = \begin{cases} 0, & \text{nếu } I(t) \geq d_0 \\ 1, & \text{nếu } I(t) \leq d_1 \\ X, & d_1 < I(t) < d_0 \end{cases} \quad (1.8)$$

Trong đó:

- d_0 , d_1 là các ngưỡng phát hiện bit, bit X có thể là bit "0" hoặc bit "1".
- Nếu tín hiệu thu được $I(t) \geq d_0$, Bob nhận bit "0".
- Nếu tín hiệu thu được $I(t) \leq d_1$, Bob nhận bit "1".



Hình 6 - Mô phỏng giải mã tín hiệu

Hình 6 vẽ gồm ba đồ thị thể hiện quá trình truyền, tiếp nhận và phát hiện bit trong một hệ thống truyền thông quang sử dụng điều chế cường độ ánh sáng. Đồ thị đầu tiên mô tả chuỗi bit nhị phân ban đầu được truyền từ nguồn phát (LED). Các bit được hiển thị theo chỉ số bit và có giá trị 0 hoặc 1, là thông tin đầu vào của hệ thống. Đồ thị thứ hai thể hiện dòng điện I_t được tạo ra tại photodiode (PD) khi nhận ánh sáng từ LED. Tín hiệu đầu ra này bao gồm hai mức biên độ đối xứng âm-dương tương ứng với hai bit (0 hoặc 1), cùng với thành phần nhiễu Gaussian có độ lệch chuẩn đã cho. Kết quả là dòng điện dao động xung quanh hai mức lý tưởng, có thể bị chênh lệch do nhiễu ngẫu nhiên, phản ánh rõ tác động của kênh truyền và

nhiễu điện tử trong quá trình nhận. Cuối cùng, đồ thị thứ ba là chuỗi bit được phát hiện sau khi so sánh dòng điện I_t với ngưỡng phát hiện (*threshold = 0*). Nếu dòng điện nhỏ hơn hoặc bằng 0 thì phát hiện là bit 1, ngược lại là bit 0. Đồ thị này cho thấy khả năng phục hồi lại chuỗi bit gốc tại phía thu, đồng thời có thể cho thấy sai số nếu nhiễu gây ra nhận sai bit.

2.1.8 Xác suất lỗi bit lượng tử - QBER

$$QBER = \frac{P_{sift}}{P_{error}} \tag{1.9}$$

Trong đó:

- $QBER$: tỷ lệ lỗi bit lượng tử.
- P_{error}: xác suất Bob nhận sai bit (nhận "0" thay vì "1" và ngược lại).
- P_{sift}: xác suất Bob chọn đúng cơ sở đo, có thể giải mã tín hiệu từ Alice.
- $P_{sift} = P_{A,B}(0,0) + P_{A,B}(0,1) + P_{A,B}(1,0) + P_{A,B}(1,1).$ (1.10)
- $P_{error} = (0,1) + P_{A,B}(1,0) + P_{A,B}(1,0).$ (1.11)

2.1.9 Xác suất lỗi bit tại Bob

$$P_{A,B}(a,0) = \frac{1}{2} Q\left(\frac{d_0 - I_a}{\sigma_N}\right) \tag{1.12}$$

$$P_{A,B}(a,1) = \frac{1}{2} Q\left(\frac{I_a - d_1}{\sigma_N}\right) \tag{1.13}$$

Trong đó:

- $P_{A,B}(a,0)$: xác suất Bob nhận nhầm bit "0" khi Alice gửi bit a.
- $P_{A,B}(a,1)$: xác suất Bob nhận nhầm bit "1" khi Alice gửi bit a.
- $Q(x)$: hàm Q phân phối chuẩn, mô tả xác suất lỗi gây ra bởi nhiễu Gauss.
- d_0, d_1: ngưỡng quyết định bit.
- I_a: tín hiệu trung bình tại bộ thu.
- σ_N: độ lệch chuẩn của nhiễu tổng hợp.

2.1.10 Tổng phương sai của nhiễu

$$\sigma_N^2 = \sigma_{sh}^2 + \sigma_b^2 + \sigma_{th}^2 \tag{1.14}$$

Trong đó:

- σ_N^2: tổng phương sai nhiễu tại bộ thu.
- σ_{sh}^2: phương sai nhiễu shot noise.
- σ_b^2: phương sai nhiễu nền (background noise).
- σ_{th}^2: phương sai nhiễu nhiệt (thermal noise).

2.1.11 Phương sai nhiễu shot noise

$$\sigma_{sn}^2 = 2q\Re \cdot \left(\frac{P}{4}\right)\rho g(\psi)T_5(\psi)H(d,\psi)B \quad (1.15)$$

Trong đó:

- σ_{sn}^2: phương sai nhiễu shot noise.
- q: điện tích electron $(1.6 \times 10^{-19}\ C)$.
- $\Re$: độ nhạy của photodetector (A/W).
- P: công suất quang nhận được.
- ρ: độ sâu điều chế.
- B: băng thông của hệ thống (Hz).

2.1.12 Phương sai nhiễu nền

$$\sigma_b^2 = 2qI_B B \quad (1.14)$$

Trong đó:

- σ_b^2: phương sai nhiễu nền.
- I_B: dòng điện nền.
- B: băng thông của hê thống.

2.1.13 Phương sai nhiễu nhiệt

$$\sigma_{th}^2 = \frac{8\pi kT_k}{G_{ol}} C_{pd} A I_2 B^2 + \frac{16\pi^2 kT_k \Gamma}{g_m} C_{pd}^2 A^2 I_3 B^2 \quad (1.15)$$

Trong đó:

- k: hằng số Boltzmann.
- T_k: nhiệt độ tuyệt đối (Kelvin).
- G_{ol}: hệ số khuếch đại mạch mở.
- Γ: hệ số nhiễu của transitor (FET).

2.2 Phân phối khóa lượng tử biến rời rạc DV-QKD

2.2.1 Giao thức BB84

Trong mật mã lượng tử, các trạng thái phân cực khác nhau của photon được sử dụng để mã hoá và giải mã. Nếu chúng ta đo phân cực của một photon thông qua hệ đo phân cực theo đường thẳng thì các kết quả đo sẽ chỉ ra rằng photon đó phân cực thẳng đứng hay nằm ngang. Hoàn toàn tương tự như vậy cho hệ phân cực chéo. Ta quy ước các ký hiệu như sau:

- $\oplus$: thiết bị đo phân cực thẳng.
- $\updownarrow$: phân cực thẳng đứng.

• ↔ : phân cực thẳng ngang.

• ⊗ : thiết bị đo phân cực chéo.

• ↖ : phân cực chéo hướng trái.

• ↗ : phân cực chéo hướng phải.

Mã hoá và giải mã lượng tử được thực hiện dựa trên trạng thái phân cực của photon. Trạng thái của các photon khi đi qua các hệ đo phân cực khác nhau như sau:

• Một photon trong hệ phân cực thẳng thì có thể là phân cực thẳng đứng hoặc ngang:

– Photon 1: ↔ ⊕ ↔

– Photon 2: ↕ ⊕ ↕

• Nếu một photon được gửi liên tiếp qua các hệ đo phân cực giống nhau thì cho kết quả không đổi:

– Photon 1: ↔ ⊕ ↔ ⊕ ↔

– Photon 2: ↕ ⊕ ↕ ⊕ ↕

• Một photon phân cực thẳng hoặc ngang nếu truyền qua hệ đo phân cực chéo sẽ cho kết quả là phân cực chéo trái hoặc phải, ví dụ: ↕ ⊗ ↖ hoặc ↕ ⊗ ↗.

• Tương tự, truyền một photon phân cực chéo qua thiết bị đo phân cực thẳng cũng cho kết quả ngẫu nhiên là thẳng đứng hoặc ngang.

Giao thức BB84 (do Bennett và Brassard giới thiệu năm 1984) dựa trên tính chất bất định và không thể sao chép được các trạng thái lượng tử. Kẻ nghe trộm trên đường truyền (Eve) không thể đọc thông tin mà không làm thay đổi các trạng thái lượng tử. Nếu phát hiện có sự nghe trộm, Alice và Bob có thể ngay lập tức hủy bỏ khóa lượng tử đó và bắt đầu lại quá trình trao đổi khóa mới, đảm bảo rằng thông tin cần bảo mật không bị ảnh hưởng. Nhờ cơ chế này, giao thức BB84 cung cấp một phương pháp an toàn để thiết lập khóa mật mã mà không cần dựa vào giả định về độ phức tạp tính toán như các phương pháp mật mã truyền thống. Đây chính là nền tảng quan trọng cho các hệ thống truyền thông lượng tử an toàn trong tương lai.

Ta quy ước:

• Alice là người gửi thông tin.

• Bob là người nhận thông tin.

• Eve là kẻ nghe trộm.

➢ Các bước của giao thức BB84 như sau:

1. Alice chọn ngẫu nhiên các photon theo cả hai hệ đo phân cực thẳng và phân cực chéo.

2. Alice ghi lại các trạng thái phân cực của các photon và gửi chúng cho Bob.

3. Bob nhận photon và đo phân cực một cách ngẫu nhiên theo hệ đo thẳng hoặc chéo. Bob ghi lại hệ đo đã sử dụng và kết quả đo. Lưu ý rằng kết quả này có thể khác Alice nếu hai người dùng hệ đo khác nhau.

4. Bob công bố công khai hệ đo mà mình đã sử dụng và tất nhiên là không công bố kết quả đo.

5. Alice đối chiếu và thông báo lại những hệ đo nào trùng với hệ đo cô đã sử dụng.

6. Alice và Bob loại bỏ các photon có hệ đo không trùng nhau. Dữ liệu còn lại sẽ được mã hóa thành chuỗi bit theo quy ước:

• ↖: 1, ↗: 0

• ↔: 1, ↕: 0

Bảng 1- Minh họa giao thức BB84

TT	Mô tả	1	2	3	4	5	6	7	8	9	10	11	12
1	Hệ đo Alice dùng	⊕	⊕	⊗	⊕	⊗	⊗	⊗	⊕	⊕	⊕	⊗	⊗
2	Kết quả Alice gửi	↑	↔	↖	↑	↖	↖	↖	↔	↑	↔	↗	↖
3a	Hệ đo Bob dùng	⊕	⊕	⊕	⊕	⊗	⊗	⊗	⊗	⊕	⊗	⊗	⊕
3b	Kết quả Bob đo	↑	↔	↔	↔	↔	↖	↖	↔	↑	↔	↖	↔
4	Bob gửi hệ đo	⊕	⊕	⊕	⊕	⊗	⊗	⊗	⊗	⊕	⊗	⊗	⊕
5	Alice gửi hệ đúng	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗
6	Chuỗi bit sau lọc	1	0	X	0	1	0	1	X	0	X	1	X

Bảng 1 minh họa các bước trong giao thức BB84 thông thường ứng với quy trình 6 bước mà tôi đã trình bày ở trên. Kết quả cuối cùng, Alice và Bob thu được một chuỗi bit làm khóa chung gồm các bit như sau:

1 0 0 1 0 1 0 1

Chi tiết về giao thức BB84 như sau:

Bước 1: Alice tạo chuỗi bit ngẫu nhiên

- Alice tạo ra chuỗi bit ngẫu nhiên có độ dài là *n.*
- Mỗi bit có thể là *0* hoặc *1*.

→ Với mỗi bit có 2 giá trị là *0* hoặc *1* cho nên với chuỗi có độ dài là n thì sẽ có 2^n khả năng chuỗi bit khác nhau.

Ví dụ với n=3, các khả năng mà Alice có thể tạo ra:

$$\begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix}$$

→ Tổng số trường hợp với n bit là 2^n.

Bước 2: Alice chọn cơ sở mã hóa cho từng bit

Với mỗi bit, Alice chọn ngẫu nhiên một trong hai cơ sở:

- Cơ sở trực chuẩn (Z-basic): { $|0\rangle$, $|1\rangle$ }.
- Cơ sở chéo (X-basic): {$|+\rangle$, $|-\rangle$ }.

Quy tắc mã hóa như sau:

Bảng 2 - Quy tắc mã hóa

Bit	Cơ sở	Trạng thái photon mã hóa
0	Z	$\|0\rangle$
1	Z	$\|1\rangle$
0	X	$\|+\rangle = \frac{\|0\rangle + \|1\rangle}{\sqrt{2}}$
1	X	$\|-\rangle = \frac{\|0\rangle - \|1\rangle}{\sqrt{2}}$

Bảng 2 là quy tắc mã hóa để ánh xạ từ bit sang qubit. Với mỗi bit có 2 giá trị là 0 hoặc 1 và có 2 cơ sở Z hoặc X nên có 4 khả năng xảy ra cho mỗi bit. Với n bit, 2^n cách chọn bit, và 2^n cách chọn cơ sở nên tổng hợp 4^n khả năng tổ hợp (bit + cơ sở). Ví dụ $n = 2$, giả sử chuỗi bit Alice chọn là [1, 0], cô có thể chọn cơ sở như sau:

Bảng 3 - Quy tắc mã hóa với n=2

Vị trí	Bit	Cơ sở	Trạng thái gửi đi
1	1	Z	$\|1\rangle$
1	1	X	$\|-\rangle$
2	0	Z	$\|0\rangle$
2	0	X	$\|+\rangle$

Bảng 3 là quy tắc mã hóa bit sang qubit với $n = 2$, có bốn trường hợp khả thi cho hai bit. Tổng quát cho **Bước 1** và **Bước 2**, ta có:

Bảng 4 - Tổng quát tạo bit và mã hóa

Index	Bit	Cơ sở	Trạng thái gửi đi
i	*0/1*	*Z/X*	$\|0\rangle, \|1\rangle, \|+\rangle, \|-\rangle$

Bảng 4 là tổng hợp ngắn gọn quy trình hai bước đầu, với hai bit 0 và 1 ứng với 2 cơ sở Z và X sẽ cho ra 4 trạng thái qubit khác nhau.

Bước 3: Bob nhận photon và đo với cơ sở ngẫu nhiên

Với mỗi photon, Bob có hai lựa chọn:

- Z hoặc X.

Kết quả đo của Bob phụ thuộc vào:

- Cơ sở mà Alice dùng để mã hóa.
- Cơ sở mà Bob dùng để đo.

Tất cả các trường hợp có thể xảy ra với từng photon như sau:

Bảng 5 - Các trường hợp xảy ra với từng photon

Bit Alice	Cơ sở Alice	Cơ sở Bob	Trạng thái truyền	Xác suất Bob đo đúng	Ghi chú
0	Z	Z	$\|0\rangle$	100%	Cùng cơ sở
1	Z	Z	$\|1\rangle$	100%	Cùng cơ sở
0	X	X	$\|+\rangle$	100%	Cùng cơ sở
1	X	X	$\|-\rangle$	100%	Cùng cơ sở
0	Z	X	$\|0\rangle$	50% ($\|+\rangle$ *hoặc* $\|-\rangle$)	Khác cơ sở
1	Z	X	$\|1\rangle$	50%	Khác cơ sở
0	X	Z	$\|+\rangle$	50% ($\|0\rangle$ *hoặc* $\|1\rangle$)	Khác cơ sở
1	X	Z	$\|-\rangle$	50%	Khác cơ sở

Bảng 5 liệt kê tất cả các trường hợp có thể xảy ra khi Bob nhận photon và đo chúng với cơ sở ngẫu nhiên. Trong đó nếu cơ sở của Bob trùng với cơ sở của Alice thì xác suất đo đúng trong điều kiện lý tưởng là 100%. Tất cả các trường hợp khác cơ sở thì xác suất đo đúng là 50%.

Bước 4: Bob và Alice so sánh cơ sở

Với mỗi photon, có 4 trường hợp kết hợp giữa cơ sở của Alice và cơ sở của Bob trong *Bảng 6* như sau:

Bảng 6 - Bob và Alice so sánh cư sở

Cơ sở Alice	Cơ sở Bob	Hành động
Z	Z	Giữ lại bit
Z	X	Loại bỏ bit
X	Z	Loại bỏ bit
X	X	Giữ lại bit

Từ *Bảng 6*, t có thể thấy có hai trường hợp giữ lại nếu cùng cơ sở và hai trường hợp bỏ đi nếu khác cơ sở.

Bước 5: Trích xuất khóa tạm thời

Nếu cơ sở của Alice = cơ sở của Bob:

- Nếu Bob đo đúng → Bit giữ nguyên.
- Nếu Bob đo sai (do nhiễu hoặc Eve) → Bit sai.

→ Trường hợp có lỗi, cần được phát hiện ở bước sau.

Bước 6: Trường hợp có kẻ nghe lén

Giả sử Eve thực hiện tấn công kiểu đo – ghi lại (intercept-resend attack), tức là cô ta chặn các photon mà Alice gửi đi, đo chúng bằng cơ sở ngẫu nhiên, sau đó gửi lại photon mới đến Bob theo kết quả đo được. Khi ba bên (Alice, Eve, Bob) chọn cơ sở đo không đồng nhất, xác suất lỗi tăng lên rõ rệt. Điều này làm xuất hiện sai khác giữa chuỗi bit của Alice và Bob, được phản ánh thông qua QBER.

Bảng 7 trình bày ma trận xác suất xảy ra lỗi trong từng trường hợp cụ thể khi có sự can thiệp của Eve, giúp đánh giá khả năng phát hiện nghe lén dựa trên các tổ hợp cơ sở đo và tỉ lệ lỗi tương ứng.

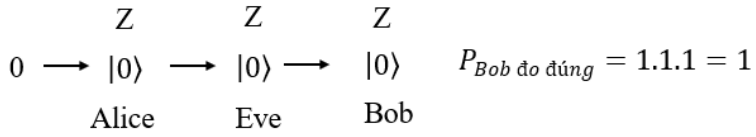
Bảng 7 - Xác suất Bob đo đúng trong trường hợp có Eve

Bit Alice	Cơ sở Alice	Cơ sở Eve	Cơ sở Bob	Xác suất Bob nhận đúng	Xác suất lỗi
0	Z	Z	Z	100%	0%
1	Z	Z	Z	100%	0%
0	X	X	X	100%	0%
1	X	X	X	100%	0%
0	Z	X	Z	50%	50%
1	Z	X	Z	50%	50%
0	X	Z	X	50%	50%
1	X	Z	X	50%	50%

0	Z	Z	X	50%	50%
1	Z	Z	X	50%	50%
0	X	X	Z	50%	50%
1	X	X	Z	50%	50%
0	Z	X	X	50%	50%
1	Z	X	X	50%	50%
0	X	Z	Z	50%	50%
1	X	Z	Z	50%	50%

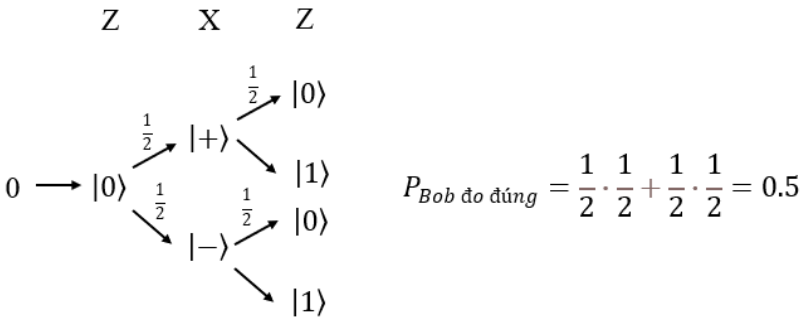
Chứng minh xác suất Bob nhận đúng trong Bảng 7:

Trường hợp 1: Alice, Bob và Eve có chung cơ sở:



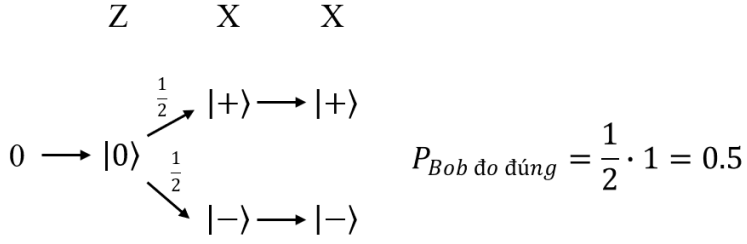
Alice mã hóa bit 0 bằng cơ sở Z → trạng thái |0⟩. Eve đo bằng cơ sở Z → nhận đúng |0⟩, sau đó gửi lại |0⟩ cho Bob. Bob cũng đo theo cơ sở Z → chắc chắn nhận lại |0⟩. Không có sai lệch do mọi bên đều dùng chung cơ sở.

Trường hợp 2: Alice, Bob có chung cơ sở và khác cơ sở với Eve:



Alice gửi |0⟩ (cơ sở Z).,Eve đo bằng cơ sở X (không phù hợp) → có xác suất một nửa nhận |+⟩ hoặc |−⟩. Sau đó Eve gửi lại photon tương ứng |+⟩ hoặc |−⟩. Bob đo photon này bằng cơ sở Z → khi đo |+⟩hoặc |−⟩ trong cơ sở Z, Bob có xác suất một nửa đo ra |0⟩. Có tổng cộng 4 khả năng tương tác (Eve nhận |+⟩/|−⟩, Bob đo |0⟩/|1⟩). Trong đó, 2 khả năng dẫn đến Bob nhận đúng.

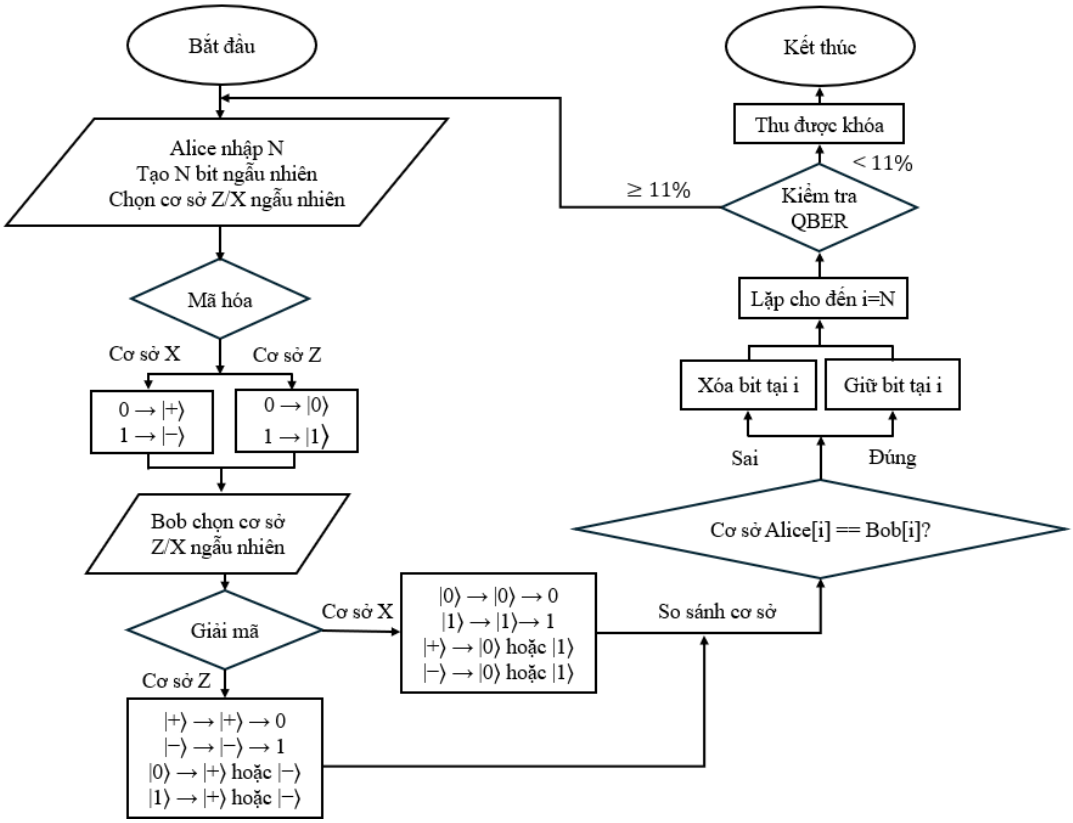
Trường hợp 3: Bob và Eve có chung cơ sở và khác cơ sở với Alice:



Alice gửi |0⟩ (cơ sở Z), Eve đo bằng cơ sở X → có xác suất một nửa nhận |+⟩ hoặc |−⟩|. Eve gửi lại |+⟩/|−⟩ cho Bob, Bob cũng đo bằng X → sẽ nhận lại đúng trạng thái |+⟩hoặc |−⟩. Nhưng vì Alice dùng cơ sở khác nên bit Bob thu được chỉ có 50% khả năng trùng với bit Alice.

Trong tấn công intercept-resend, xác suất Bob nhận đúng bit của Alice phụ thuộc vào sự trùng khớp cơ sở giữa ba bên. Nếu Alice, Eve và Bob cùng chọn một cơ sở, Bob luôn nhận đúng với xác suất 100%. Nếu Eve chọn sai cơ sở, kể cả khi Alice và Bob trùng cơ sở, việc đo sai của Eve làm giảm xác suất Bob nhận đúng xuống còn 50%. Trường hợp Bob và Eve trùng cơ sở, nhưng khác với Alice, thì kết quả Bob nhận cũng chỉ đúng với xác suất 50% do photon đã bị biến đổi bởi việc đo sai cơ sở trước đó của Eve.

Lưu đồ thuật toán lập trình cho giao thức BB84 thông thường:



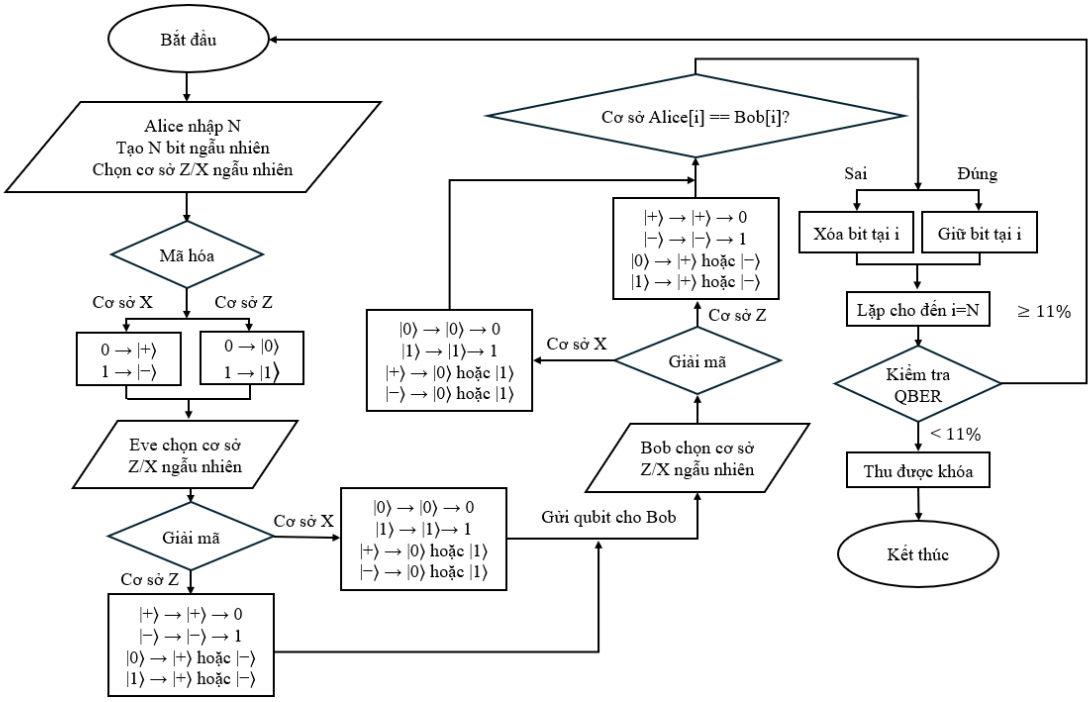
Hình 7 - Lưu đồ thuật toán BB84 thông thường

Hình 7 mô tả thuật toán BB84, là một giao thức phân phối khóa lượng tử giữa hai bên Alice và Bob. Quá trình bắt đầu khi Alice nhập vào một số nguyên N – số bit mà cô muốn gửi, sau đó tạo ra N bit nhị phân ngẫu nhiên cùng với N cơ sở ngẫu nhiên được chọn từ hai loại: cơ sở Z (gồm các trạng thái |0⟩, |1⟩) và cơ sở X (gồm các trạng thái |+⟩, |−⟩). Mỗi bit nhị phân được mã hóa thành một qubit tương ứng dựa trên cơ sở đã chọn. Ví dụ, nếu chọn cơ sở Z thì bit 0 được mã hóa thành |0⟩, bit 1 thành |1⟩; còn nếu chọn cơ sở X thì bit 0 thành |+⟩ và bit 1 thành |−⟩.

Sau đó, các qubit được gửi đến Bob thông qua kênh lượng tử. Bob cũng chọn ngẫu nhiên cơ sở Z hoặc X để đo từng qubit. Nếu cơ sở đo của Bob trùng với cơ sở mà Alice đã dùng để mã hóa thì kết quả đo sẽ chính xác và Bob có thể thu được đúng bit. Ngược lại, nếu cơ sở không trùng, kết quả đo sẽ là ngẫu nhiên và không có giá trị sử dụng. Tiếp theo, Alice và Bob công khai so sánh các cơ sở đã sử dụng tại từng vị trí. Nếu cơ sở trùng nhau, họ giữ lại bit tại vị trí đó; nếu không trùng, bit bị loại bỏ. Quá trình này được lặp lại cho đến khi xét hết N vị trí ban đầu. Từ các bit giữ lại, họ sẽ hình thành một chuỗi bit chung.

Để kiểm tra tính an toàn, một phần trong các bit giữ lại sẽ được sử dụng để tính tỷ lệ lỗi lượng tử (QBER – Quantum Bit Error Rate). Nếu QBER nhỏ hơn ngưỡng cho phép (thường là 11%), họ chấp nhận chuỗi bit còn lại làm khóa chung bí mật. Nếu QBER vượt quá ngưỡng, quá trình bị hủy vì nghi ngờ có sự nghe lén. Quá trình kết thúc khi cả hai bên có được khóa lượng tử dùng cho mã hóa thông tin an toàn.

Lưu đồ thuật toán lập trình cho giao thức BB84 khi có Eve:



Hình 8 - Lưu đồ thuật toán BB84 khi có Eve

Hình 8 là sơ đồ mô tả quy trình phân phối khóa lượng tử BB84 với sự tham gia của kẻ nghe lén Eve. Alice tạo N bit ngẫu nhiên, mã hóa bằng cơ sở Z hoặc X rồi gửi đến Bob. Eve chặn và đo các qubit với cơ sở ngẫu nhiên, làm thay đổi trạng thái nếu chọn sai cơ sở. Bob tiếp tục đo qubit với cơ sở ngẫu nhiên. Sau đó, Alice và Bob so sánh cơ sở; nếu trùng thì giữ bit, không thì bỏ. Cuối cùng, họ tính QBER: nếu ≤11% thì thu được khóa; nếu >11% thì nghi ngờ bị nghe lén và loại bỏ khóa.

2.2.2 So sánh DV-QKD và CV-QKD

Trong lĩnh vực phân phối khóa lượng tử (Quantum Key Distribution – QKD), hai phương pháp phổ biến là QKD biến rời rạc (Discrete-Variable QKD – DV-QKD) và QKD biến liên tục (Continuous-Variable QKD – CV-QKD). DV-QKD sử dụng các trạng thái lượng tử rời rạc như phân cực photon để mã hóa thông tin nhị phân (ví dụ 0 và 1), điển hình là giao thức BB84. Quá trình truyền và nhận đòi hỏi thiết bị tạo và phát hiện photon đơn với độ chính xác cao, do đó thường phức tạp và chi phí lớn, nhưng lại đảm bảo độ bảo mật rất mạnh, đặc biệt phù hợp trong các hệ thống cần độ tin cậy cao hoặc truyền ở khoảng cách xa.

Trong khi đó, CV-QKD mã hóa thông tin lên các biến liên tục như biên độ và pha của sóng ánh sáng laser, cho phép sử dụng các thiết bị thông thường như homodyne và heterodyne detector. Nhờ vậy, CV-QKD có thể đạt tốc độ truyền cao và dễ tích hợp vào các mạng viễn thông sợi quang hiện có. Tuy nhiên, do tín hiệu liên tục dễ bị nhiễu và suy hao, CV-QKD thường nhạy cảm hơn trong môi trường truyền dẫn thực tế, đặc biệt là ở khoảng cách dài. Tóm lại, DV-QKD thiên về độ bảo mật cao và triển khai chuyên biệt, trong khi CV-QKD phù hợp với hạ tầng quang học hiện đại và nhu cầu truyền khóa tốc độ cao trong khoảng cách ngắn đến trung bình.

2.3 Phân phối Gamma – Gamma

2.3.1 Phân phối Gamma

Phân phối Gamma là một phân phối xác suất liên tục, được sử dụng rộng rãi trong thống kê, lý thuyết xác suất và các mô hình vật lý, kỹ thuật, tài chính. Nó mô tả thời gian chờ đợi cho đến khi xảy ra một số lượng sự kiện nhất định trong một quá trình Poisson. Một biến ngẫu nhiên X có phân phối Gamma nếu hàm mật độ xác suất (PDF) của nó là:

$$f_X(x;\alpha,\theta) = \frac{1}{\Gamma(\alpha)\theta^{\alpha}} x^{\alpha-1} e^{-x/\theta}, \qquad x > 0 \tag{1.16}$$

Trong đó:

- $\alpha > 0$: tham số hình dạng (shape parameter).
- $\theta > 0$: tham số tỷ lệ (scale parameter).
- $\Gamma(\alpha)$: là hàm Gamma, với $\Gamma(n) = (n-1)!$ nếu n là số nguyên dương.

Tính chất:

- Kỳ vọng (mean): $E[x] = \alpha\theta$.
- Phương sai (variance): $Var[x] = \alpha\theta^2$.
- Nếu $\alpha = 1$, thì phân phối Gamma trở thành phân phối mũ (Exponential).
- Nếu α là số nguyên, thì phân phối Gamma trở thành phân phối Erlang.

Ứng dụng:

- Mô tả thời gian giữa các sự kiện trong một tiến trình Poisson.
- Mô hình hóa nhiễu và fading trong truyền thông không dây quang học.
- Trong kỹ thuật, dùng để mô phỏng tuổi thọ thiết bị, thời gian chờ đợi, hoặc phân phối năng lượng nhận được.

2.3.2 Phân phối Gamma-Gamma

Giả sử ta có:

- $X \sim Gamma(\alpha, \theta_1)$: biến ngẫu nhiên Gamma với tham số hình dạng (shape) α và tham số tỷ lệ (scale) θ_1.
- $Y \sim Gamma(\beta, \theta_2)$: biến ngẫu nhiên Gamma khác, độc lập với X với tham số β, θ_2.

Khi đó:

$$I = X \cdot Y \tag{1.17}$$

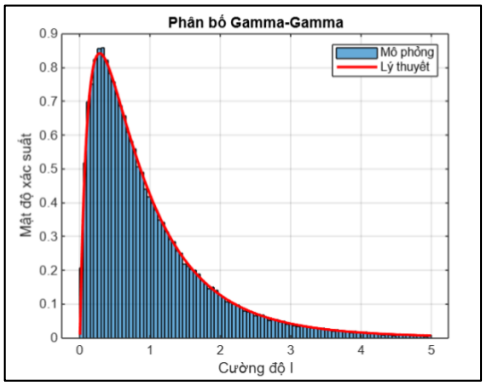
là một biến ngẫu nhiên có phân phối Gamma-Gamma.

Phân phối Gamma-Gamma mô tả biến thiên của cường độ tín hiệu nhận được do ảnh hưởng của nhiễu loạn đa quy mô (tức là gồm cả nhiễu loạn lớn và nhỏ) trong môi trường truyền quang. Nó được hình thành từ sự kết hợp của hai phân phối Gamma: một phân phối mô hình hóa nhiễu loạn quy mô nhỏ (scattering nhỏ, dao động ngắn hạn); một phân phối mô hình hóa nhiễu loạn quy mô lớn (scattering lớn, dao động dài hạn).. Biểu thức hàm mật độ xác suất PDF:

$$f_I(I) = \frac{2(\alpha\beta)^{\frac{\alpha+\beta}{2}}}{\Gamma(\alpha)\Gamma(\beta)} I^{\frac{\alpha+\beta}{2}-1} K_{\alpha-\beta}\left(2\sqrt{\alpha\beta I}\right) \tag{1.18}$$

Trong đó:

- I: cường độ tín hiệu nhận được.
- α, β: tham số mô hình hóa mức độ nhiễu loạn khí quyển.
- $\Gamma(\cdot)$: hàm Gamma.
- $K_v(\cdot)$: hàm Bessel loại hai, bậc v.



Hình 9 – Hàm PDF của phân bố Gamma-Gamma

Hình 9 mô tả đồ thị hiển thị sự so sánh giữa kết quả mô phỏng (cột histogram) và hàm mật độ xác suất lý thuyết (đường cong màu đỏ) của phân bố Gamma-Gamma với các tham số $\alpha = 3$ và $\beta = 3$. Các cột màu xanh thể hiện histogram của mẫu mô phỏng, được sinh ra bằng cách lấy tích của hai biến ngẫu nhiên Gamma độc lập. Đây là kỹ thuật mô phỏng Monte Carlo nhằm ước lượng mật độ xác suất thực nghiệm của kênh fading. Đường màu đỏ là giá trị hàm mật độ xác suất (PDF) Gamma-Gamma được tính theo công thức lý thuyết. Đường cong này mô tả xác suất tương đối để tín hiệu có giá trị cường độ *I* cụ thể trong môi trường bị fading. Kết quả cho thấy hai đường (mô phỏng và lý thuyết) khớp rất sát nhau, chứng minh rằng quá trình mô phỏng là chính xác. Điều này xác nhận rằng phân bố của tích hai biến Gamma thật sự tuân theo phân bố Gamma-Gamma.

Trong mô hình truyền thông quang học như FSO hay VLC, ta thường gặp fading nhiễu loạn hai tầng (two-scale turbulence): Gamma thứ nhất đại diện cho nhiễu loạn quy mô nhỏ (ví dụ: dao động ngắn hạn do hiện tượng nhiễu xạ - scattering). Gamma thứ hai đại diện cho nhiễu loạn quy mô lớn (ví dụ: thay đổi mật độ không khí hoặc áp suất gây hiệu ứng khúc xạ). Khi kết hợp lại, ta thu được Gamma-Gamma – một mô hình xác suất có khả năng mô tả đầy đủ hiệu ứng fading tổng hợp trong các môi trường phức tạp.

2.4 Kết luận chương

Các nội dung lý thuyết đã được hệ thống một cách chi tiết và logic, giúp người đọc hiểu rõ các khái niệm cốt lõi như CV-QKD, DV-QKD, và fading Gamma-Gamma. Những kiến thức này là cơ sở để xây dựng và đánh giá mô hình mô phỏng trong chương tiếp theo.

CHƯƠNG 3. TRIỂN KHAI MÔ PHỎNG

Trong chương này, hệ thống mô phỏng giao thức BB84 được triển khai dưới nhiều điều kiện kênh truyền khác nhau. Các công cụ như Python, MATLAB và Streamlit được sử dụng để đánh giá ảnh hưởng của nhiễu và sự can thiệp của kẻ nghe lén đến chất lượng truyền khóa lượng tử.

3.1 Các ngôn ngữ, công cụ mô phỏng

Trong quá trình nghiên cứu và phát triển mô phỏng hệ thống truyền thông lượng tử, việc lựa chọn công cụ phù hợp đóng vai trò rất quan trọng nhằm đảm bảo tính chính xác, trực quan và linh hoạt. Trong chương này, chúng tôi giới thiệu các công cụ chính được sử dụng gồm MATLAB, Python, Streamlit, HTML và CSS.

MATLAB là một môi trường tính toán số và lập trình mạnh mẽ, đặc biệt hữu ích trong việc mô phỏng các mô hình vật lý, xử lý tín hiệu, và trực quan hóa kết quả. Với các thư viện hỗ trợ như Communications Toolbox, MATLAB giúp xây dựng và phân tích mô hình hệ thống truyền thông lượng tử một cách hiệu quả.

Python là một ngôn ngữ lập trình mã nguồn mở rất phổ biến trong lĩnh vực khoa học và kỹ thuật nhờ vào cú pháp đơn giản, dễ đọc, và khả năng mở rộng mạnh mẽ thông qua các thư viện. Trong các mô phỏng hệ thống lượng tử như QKD (Quantum Key Distribution), Python đóng vai trò là công cụ lập trình chính để xây dựng thuật toán, xử lý dữ liệu và tính toán các thông số quan trọng như QBER (Quantum Bit Error Rate), xác suất lỗi bit, hiệu suất kênh truyền, v.v. Các thư viện như NumPy và SciPy hỗ trợ tính toán số học, đại số tuyến tính, và xử lý tín hiệu hiệu quả, trong khi Matplotlib cung cấp công cụ vẽ đồ thị trực quan để hiển thị kết quả mô phỏng và phân tích dữ liệu đầu ra.

Streamlit là một thư viện mã nguồn mở trên nền tảng Python, cho phép xây dựng giao diện người dùng dạng web một cách nhanh chóng và trực quan mà không cần kiến thức chuyên sâu về phát triển web. Với cú pháp đơn giản và tích hợp chặt chẽ với Python, Streamlit đặc biệt phù hợp cho các nhà nghiên cứu, sinh viên và kỹ sư khi muốn tạo các công cụ mô phỏng hoặc trình bày kết quả phân tích một cách sinh động và dễ sử dụng. Trong ngữ cảnh mô phỏng các hệ thống truyền thông hoặc bảo mật lượng tử, Streamlit cho phép người dùng dễ dàng tạo ra các ứng dụng trình diễn mô phỏng, nơi giao diện người dùng có thể tương tác trực tiếp với mô hình. Cụ thể, người dùng có thể linh hoạt điều chỉnh các thông số đầu vào như số lượng bit cần truyền, khoảng cách giữa các thiết bị, mức độ nhiễu, hoặc hệ số fading,… thông qua các thành phần giao diện như thanh trượt, hộp nhập liệu hoặc nút bấm. Sau khi điều chỉnh các tham số, mô phỏng có thể được thực hiện chỉ bằng một cú nhấp chuột, giúp đơn giản hóa quy trình thực nghiệm và kiểm tra giả thuyết. Quan trọng hơn, kết quả mô phỏng được cập nhật và hiển thị ngay lập tức, cho phép người dùng quan sát trực tiếp sự thay đổi của các đại lượng như tỷ lệ lỗi bit lượng tử (QBER), công suất thu nhận, hoặc các biểu đồ thể hiện tín hiệu truyền – nhận. Chính khả năng phản hồi tức thì và sự thân thiện với người dùng giúp Streamlit trở thành một công cụ hiệu quả trong việc tăng tính tương tác và

trực quan hóa mô hình, phục vụ tốt cho cả mục đích học tập, trình bày học thuật cũng như nghiên cứu chuyên sâu.

Ngoài ra, HTML (HyperText Markup Language) và CSS (Cascading Style Sheets) cũng đóng vai trò quan trọng trong việc tạo và tùy chỉnh giao diện người dùng cho các ứng dụng mô phỏng chạy trên trình duyệt web. HTML đảm nhận việc định nghĩa cấu trúc nội dung của trang (ví dụ như tiêu đề, nút bấm, khung hiển thị kết quả...), trong khi CSS đảm nhiệm việc định dạng kiểu dáng, màu sắc, bố cục và hiệu ứng hiển thị. Sự kết hợp giữa HTML, CSS và các công cụ như Streamlit giúp xây dựng giao diện thân thiện, dễ sử dụng, phù hợp với nhiều đối tượng người dùng, từ sinh viên đến nhà nghiên cứu.

3.2 Mô phỏng QBER cho hệ thống CV QKD – VLC

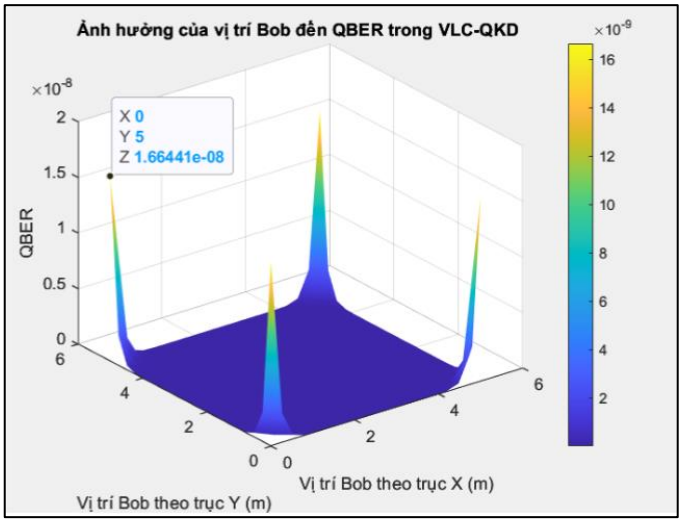
Mô phỏng lại kết quả của bài báo được thực hiện như sơ đồ Hình 1 với các tham số như sau:

Tham số	Ký hiệu	Giá trị
Kích thước phòng	*room_size*	5×5 m
Vị trí LED (giữa trần)	*LED_pos*	(2.5,2.5,3) m
Vị trí của Bob	*Bob_x, Bob_y*	[0,5] m
Công suất phát của LED	P_t	0.85 W
Diện tích PD	A	$1.10^{-4}\ m^2$
Góc bán công suất của LED	$\phi_{1/2}$	60^0
Góc thu nhận tối đa của PD	Ψ_c	60^0
Chỉ số khúc xạ của thấu kính	n	1.5
Băng thông của hệ thống	B	150 MHz
Điện tích electron	q	$1.6 \times 10^{-19}\ C$
Hằng số Boltzmann	k_B	$1.38 \times 10^{-23}\ J/K$
Nhiệt độ tuyệt đối	T_k	300 K
Độ nhạy của PD	R_e	0.54 A/W
Điện dung của PD	C_{pd}	$112 \times 10^{-12}\ F\ /\ cm^2$
Hệ số nhiễu FET	Γ	1.5
Hệ số truyền dẫn FET	g_m	$30 \times 10^{-3}\ S$
Độ sâu điều chế	ρ	0.5

Hình 10 thể hiện kết quả mô phỏng phân bố tỷ lệ lỗi bit lượng tử (QBER) trong một không gian mô phỏng dạng phòng kín, có kích thước 5×5 mét. Trong mô hình này, nguồn phát ánh sáng (LED) được đặt cố định ở vị trí chính giữa trần nhà, tức tại tọa độ trung tâm theo mặt phẳng ngang. Quá trình mô phỏng được thực hiện dựa trên các công thức tính QBER đã được trình bày chi tiết trong các phần trước của đồ án. Từ kết quả mô phỏng, có thể dễ dàng quan sát rằng các điểm ở bốn góc phòng – là những vị trí xa nhất so với nguồn phát LED – đều có giá trị

QBER cao nhất. Điều này hoàn toàn hợp lý về mặt vật lý, bởi tại các điểm này, tín hiệu thu nhận bị suy giảm mạnh do cả khoảng cách xa và góc tới lớn, làm giảm công suất thu và tăng xác suất đo sai. Ngược lại, các điểm gần trung tâm phòng – nơi LED được treo phía trên – có QBER thấp hơn rõ rệt do nhận được cường độ tín hiệu mạnh và góc thu nhỏ. Mô phỏng này không chỉ giúp trực quan hóa sự phân bố không gian của QBER trong môi trường truyền thực tế, mà còn có ý nghĩa kiểm chứng kết quả của các nghiên cứu trước đó. Cụ thể, kết quả thu được từ mô phỏng đã tái khẳng định kết luận của bài báo khoa học tham khảo, cho thấy rằng khu vực xa nguồn phát và gần biên phòng luôn là nơi xảy ra suy giảm hiệu suất truyền khóa lượng tử, từ đó làm tăng nguy cơ lỗi lượng tử và giảm mức độ bảo mật.

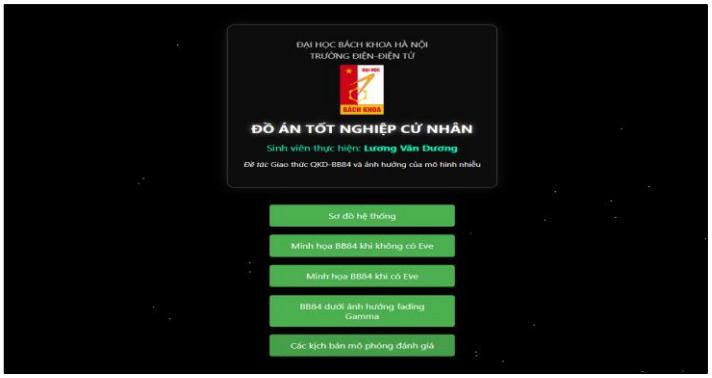
Thông qua hình ảnh trực quan như *Hình 10*, có thể nhận thấy rõ tầm quan trọng của việc bố trí vị trí người dùng và nguồn phát ánh sáng hợp lý trong các hệ thống QKD dựa trên ánh sáng nhìn thấy hoặc truyền thông quang trong phòng, nhằm đảm bảo chất lượng và độ tin cậy của quá trình phân phối khóa lượng tử.



Hình 10 - Kết quả mô phỏng

3.3 Mô phỏng giao thức BB84

Hình 11 là giao diện trang chủ của chương trình, để người dùng có thể tương tác hiệu quả với hệ thống mô phỏng, việc xây dựng một giao diện trực quan và thân thiện là vô cùng cần thiết.



Hình 11 - Giao diện trang chủ của chương trình

Chức năng đầu tiên của ứng dụng mô phỏng là tái hiện quá trình phân phối khóa lượng tử dựa trên giao thức BB84 – giao thức phân phối khóa lượng tử đầu tiên và phổ biến nhất hiện nay. Giao thức này được đề xuất bởi Charles Bennett và Gilles Brassard vào năm 1984, đặt nền móng cho lĩnh vực mật mã lượng tử hiện đại. BB84 hoạt động dựa trên các nguyên lý cơ bản của cơ học lượng tử như tính không xác định của phép đo, tính không sao chép được trạng thái lượng tử (no-cloning theorem) và tính rối lượng tử để đảm bảo an toàn tuyệt đối cho việc chia sẻ khóa bí mật giữa hai bên (Alice và Bob), ngay cả khi có sự xuất hiện của kẻ nghe lén (Eve). Trong mô phỏng này, người dùng có thể quan sát trực quan các bước diễn ra trong quá trình truyền và lọc khóa, bao gồm:

- Việc Alice tạo ngẫu nhiên các bit và chọn cơ sở phân cực tương ứng,
- Bob chọn ngẫu nhiên cơ sở đo,
- Quá trình so sánh cơ sở đo giữa Alice và Bob,
- Loại bỏ các bit không trùng cơ sở,
- Và thu được chuỗi khóa lượng tử chung.

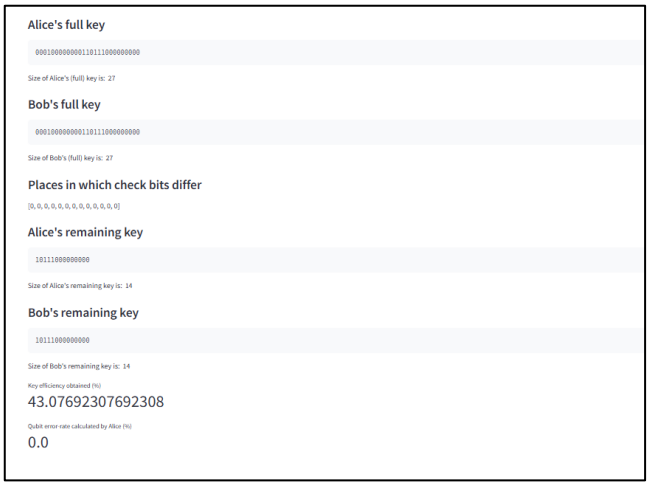
Thông qua giao diện mô phỏng, người dùng có thể tùy chọn số lượng bit truyền, theo dõi từng bước của giao thức, đồng thời trực quan hóa các thành phần như trạng thái photon, cơ sở đo, kết quả đo và chuỗi bit cuối cùng sau khi lọc. Việc mô phỏng BB84 theo cách trực quan như vậy không chỉ giúp người học hiểu sâu hơn về nguyên lý hoạt động của giao thức mà còn là công cụ giảng dạy hữu ích trong các khóa học về mật mã lượng tử và an toàn thông tin thế hệ mới.

Hình 12 - Mô phỏng giao thức BB84

Hình 12 được sử dụng nhằm mục đích minh họa trực quan toàn bộ quá trình hoạt động của giao thức phân phối khóa lượng tử BB84. Việc sử dụng sơ đồ này giúp người đọc dễ dàng hình dung và nắm bắt được các bước thực hiện cốt lõi trong giao thức, đặc biệt là với những người mới tiếp cận lĩnh vực truyền thông lượng tử. Thông qua hình vẽ, các giai đoạn quan trọng như: Alice khởi tạo chuỗi bit ngẫu nhiên, lựa chọn ngẫu nhiên giữa hai cơ sở đo (Z hoặc X), mã hóa các bit thành các trạng thái qubit tương ứng và gửi qua kênh lượng tử cho Bob; sau đó

Bob cũng chọn cơ sở đo ngẫu nhiên để đo các qubit nhận được; cuối cùng là quá trình trao đổi thông tin công khai để xác định vị trí có cơ sở đo trùng khớp và trích xuất khóa chung – đều được mô tả trực quan, rõ ràng và mạch lạc. Hình ảnh này đóng vai trò như một bản đồ tổng quát, giúp người đọc dễ dàng theo dõi luồng thông tin cũng như logic xử lý của giao thức BB84 từ đầu đến cuối.

Tuy nhiên, để hiểu rõ hơn về cơ chế hoạt động chi tiết của giao thức BB84 cũng như kiểm nghiệm tính hiệu quả của nó trong điều kiện lý tưởng, tôi đã xây dựng một mô phỏng toàn diện. Mô phỏng này tái hiện toàn bộ quá trình tạo và phân phối khóa lượng tử giữa hai bên tham gia là Alice và Bob trong một môi trường hoàn toàn lý tưởng, tức là không có nhiễu từ môi trường truyền và không có sự can thiệp của bên thứ ba như kẻ nghe lén Eve. Mục tiêu chính của mô phỏng là đánh giá khả năng đồng thuận khóa giữa Alice và Bob – cụ thể là đảm bảo rằng sau quá trình truyền và xử lý thông tin (bao gồm việc so sánh cơ sở đo và loại bỏ các bit không phù hợp), cả hai bên có thể thu được một chuỗi khóa trùng khớp tuyệt đối. Đây là bước quan trọng để xác nhận độ tin cậy và tính khả thi của giao thức BB84 trong điều kiện chuẩn trước khi tiến hành các mô phỏng phức tạp hơn có xét đến tấn công hoặc nhiễu.



Hình 13 - Kết quả mô phỏng giao thức BB84

Hình 13 là kết quả mô phỏng giao thức BB84, sau khi thực hiện quá trình chuẩn hóa và truyền qubit theo giao thức BB84 trong điều kiện lý tưởng, cả hai bên Alice và Bob đều thu được khóa lượng tử đầy đủ giống hệt nhau là: *000100000000110111000000000*, với độ dài *27* bit. Để kiểm tra tính toàn vẹn của quá trình truyền, một phần của khóa được chọn ngẫu nhiên làm bit kiểm tra và được chia sẻ công khai. Kết quả cho thấy không có bất kỳ sai khác nào giữa các bit kiểm tra, cụ thể là: *[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]*, cho thấy hệ thống không gặp lỗi trong bước truyền và đo đạc qubit. Sau khi loại bỏ phần bit kiểm tra, khóa bí mật cuối cùng được giữ lại của cả Alice và Bob là: *10111000000000*, có độ dài *14* bit. Từ đó, hiệu suất sử dụng khóa được tính toán là khoảng *43.08%*, tức chỉ khoảng *43%* số qubit ban đầu thực sự được sử dụng để tạo thành khóa cuối cùng. Đồng thời, tỷ lệ lỗi trên qubit (Qubit Error Rate – QBER) đạt *0.0%*, khẳng định rằng hệ thống hoạt động hoàn toàn lý tưởng trong mô phỏng này.

Chức năng tiếp theo trong giao diện mô phỏng là mô phỏng giao thức BB84 khi có sự tham gia của kẻ nghe lén – Eve, giúp người dùng hiểu rõ hơn về cơ chế bảo mật lượng tử cũng như khả năng phát hiện xâm nhập trong quá trình phân phối khóa. Trong thế giới cổ điển, một kẻ nghe lén có thể sao chép thông tin mà không để lại dấu vết. Tuy nhiên, trong cơ học lượng tử – đặc biệt là trong BB84 – mọi phép đo đều ảnh hưởng đến trạng thái lượng tử ban đầu. Do đó, khi Eve cố gắng nghe lén các photon được truyền từ Alice đến Bob bằng cách đo phân cực và gửi lại photon khác, cô ta vô tình làm thay đổi trạng thái của photon và gây ra lỗi trong chuỗi khóa mà Bob thu được. Sự khác biệt giữa chuỗi bit của Alice và Bob sẽ tăng lên, làm xuất hiện tỷ lệ lỗi bit lượng tử (QBER) cao bất thường – đây là dấu hiệu trực tiếp cho thấy hệ thống đang bị nghe lén. Trong mô phỏng này, người dùng có thể:

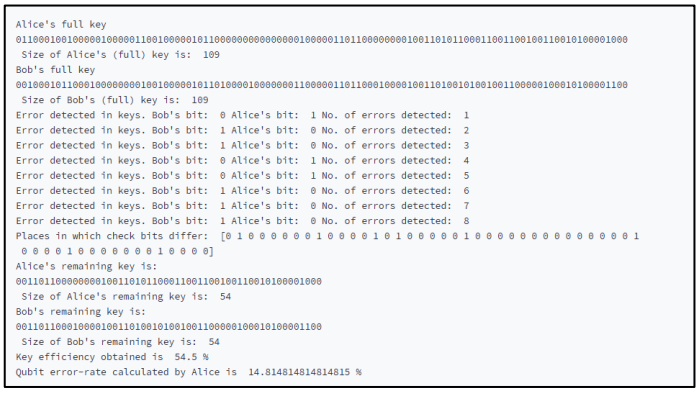
- Quan sát cách Eve chọn ngẫu nhiên cơ sở đo để chặn và đo photon của Alice.
- Xem kết quả phép đo của Eve và photon mới mà cô ta gửi tới Bob.
- So sánh chuỗi bit của Alice và Bob sau khi lọc để tính toán QBER.
- Từ đó đưa ra quyết định: tiếp tục sử dụng khóa hay hủy bỏ quá trình truyền nếu phát hiện có nghe lén.

Hình 14 là đồ họa mô phỏng giao thức BB84 khi có Eve, thông qua mô phỏng tương tác này, người dùng có thể thấy rõ ràng lý do tại sao BB84 an toàn trước các cuộc tấn công nghe lén chủ động – một trong những lợi thế quan trọng nhất của mật mã lượng tử so với mật mã cổ điển. Đây không chỉ là bài học minh họa sinh động cho tính chất vật lý của lượng tử mà còn là bằng chứng trực quan về khả năng phát hiện và phòng tránh xâm nhập của BB84 trong thực tế.

Hình 14 - Mô phỏng giao thức BB84 khi có Eve

Sau khi đã trình bày phần minh họa trực quan bằng hình ảnh để giúp người đọc hình dung rõ hơn về cách thức hoạt động của giao thức BB84 khi có Eve, tôi tiếp tục thực hiện bước mô phỏng chi tiết. Mục tiêu của mô phỏng này là đánh giá

mức độ ảnh hưởng thực tế của việc tấn công nghe lén đến quá trình phân phối khóa lượng tử giữa hai bên hợp pháp là Alice và Bob. Kết quả thu được cho thấy rõ ràng rằng, khi Eve can thiệp vào kênh truyền, sự đồng thuận khóa giữa Alice và Bob bị suy giảm đáng kể.



Hình 15 - Mô phỏng khi có Eve

Hình 15 là kết quả mô phỏng trong điều kiện có sự can thiệp của kẻ nghe lén (Eve), quá trình phân phối khóa lượng tử giữa Alice và Bob không còn lý tưởng như trước. Sau khi hoàn tất quá trình truyền và đo đạc, cả hai bên thu được khóa lượng tử ban đầu có độ dài 109 bit, nhưng không hoàn toàn trùng khớp. Cụ thể, khóa đầy đủ của Alice là:

"01100010010000010000011001000001011000000000000001000001101100000000100110101100011001100110010100001000"

Trong khi đó, khóa đầy đủ của Bob là:

"00100010110001000000000100100000101101000010000000110000011011000100001001101001010010011000001000101000001100"

Trong quá trình kiểm tra, hệ thống phát hiện có tổng cộng 8 lỗi, xảy ra do sự khác biệt giữa các bit tương ứng trong khóa của Alice và Bob. Những lỗi này là chỉ dấu cho thấy có khả năng bị tấn công trong quá trình truyền thông lượng tử. Một phần khóa được chọn ngẫu nhiên để làm bit kiểm tra nhằm phát hiện sự can thiệp. Kết quả cho thấy có nhiều vị trí bit kiểm tra bị sai lệch giữa hai bên, được phản ánh qua mảng so sánh như sau:

"[0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]"

Sau khi loại bỏ các bit kiểm tra, phần còn lại được dùng làm khóa bí mật cuối cùng. Alice thu được khóa:

"00110110000000010011010110001100110010011001010000 1000"

Và Bob thu được khóa:

"00110110001000010011010010100100110000010001010000 1100"

Cả hai khóa đều có độ dài 54 bit, cho thấy hiệu suất sử dụng khóa (Key Efficiency) đạt khoảng *54.5%.* Tuy nhiên, do có sự khác biệt trong khóa cuối cùng, tỷ lệ lỗi trên qubit (Qubit Error Rate – QBER) do Alice tính toán được là *14.81%*,

một con số cho thấy hệ thống đã bị ảnh hưởng đáng kể, phù hợp với giả thuyết có sự tấn công từ phía Eve. Đây là cơ sở để các bên phát hiện nguy cơ mất an toàn thông tin và có thể hủy bỏ quá trình trao đổi khóa nhằm đảm bảo bảo mật lượng tử.

3.4 Mô phỏng giao thức BB84 dưới ảnh hưởng của fading Gamma-Gamma

Trong các hệ thống phân phối khóa lượng tử (QKD) thực tế như QKD sử dụng truyền dẫn ánh sáng khả kiến (VLC) hoặc kênh quang tự do (FSO), tín hiệu lượng tử không chỉ chịu ảnh hưởng bởi nhiễu nền hay thiết bị mà còn bị tác động mạnh bởi nhiễu loạn môi trường truyền. Một trong những mô hình phổ biến dùng để mô tả nhiễu loạn cường độ ánh sáng trong các kênh truyền quang là fading Gamma-Gamma.

Fading Gamma-Gamma là một mô hình xác suất mô tả sự biến đổi ngẫu nhiên của cường độ tín hiệu do nhiễu loạn không khí, dao động nhiệt, bụi, khói hoặc sự thay đổi chỉ số khúc xạ. Nó đặc biệt phù hợp với môi trường truyền dẫn có tính chất nhiễu mạnh và không đồng nhất, như trong các hệ thống QKD ngoài trời, hoặc hệ thống QKD trong môi trường không kiểm soát được hoàn toàn về ánh sáng. Trong chức năng mô phỏng này, hệ thống sẽ mô phỏng:

- Sự biến động của tín hiệu ánh sáng mang photon lượng tử theo phân phối Gamma-Gamma.
- Ảnh hưởng của fading đến độ chính xác khi Bob đo tín hiệu.
- Sự thay đổi QBER (Quantum Bit Error Rate) theo thời gian hoặc theo cường độ fading.

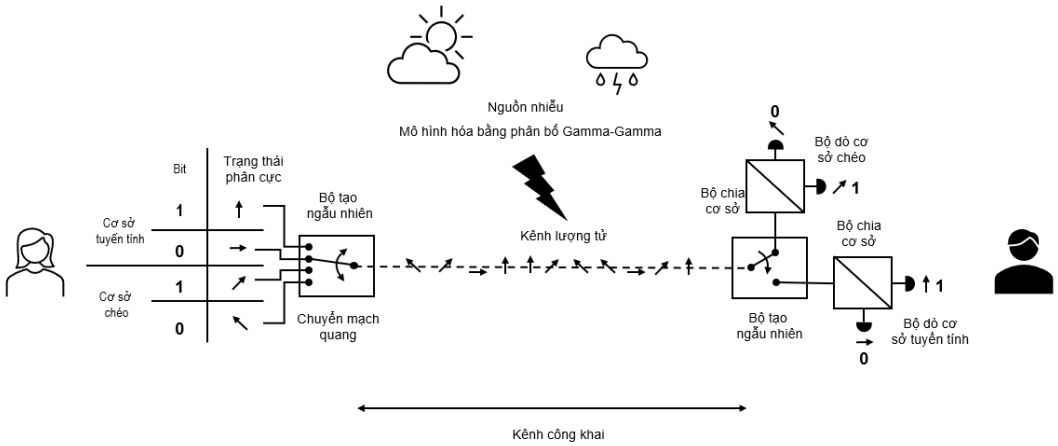
Người dùng có thể tùy ý chỉnh sửa các tham số của mô hình fading Gamma-Gamma như:

- Các tham số hình dạng (α, β) – đặc trưng cho độ mạnh/yếu của nhiễu loạn.
- Số lượng photon truyền và điều kiện nhận.

Mô phỏng này cung cấp một góc nhìn thực tiễn và sâu sắc về cách mà điều kiện kênh truyền có thể ảnh hưởng đến tính bảo mật và độ tin cậy của giao thức BB84, từ đó nhấn mạnh tầm quan trọng của việc thiết kế giao thức lượng tử phù hợp với môi trường vật lý cụ thể.

Hình 16 mô tả quy trình hoạt động của giao thức phân phối khóa lượng tử BB84 trong môi trường truyền dẫn có ảnh hưởng của nhiễu, được mô hình hóa bằng phân bố Gamma-Gamma. Trong mô hình này, bên gửi (Alice) sử dụng một bộ tạo số ngẫu nhiên để lựa chọn cơ sở mã hóa (tuyến tính hoặc chéo), sau đó ánh xạ mỗi bit nhị phân thành một trạng thái phân cực tương ứng. Các trạng thái này được truyền qua chuyển mạch quang và đi vào kênh lượng tử dưới dạng photon mang thông tin. Trong quá trình truyền, photon chịu ảnh hưởng của môi trường, bao gồm các yếu tố như ánh sáng mặt trời, sương mù, hoặc dao động không khí. Những nhiễu loạn này làm thay đổi cường độ tín hiệu và được mô hình hóa thông qua phân bố Gamma-Gamma – một mô hình xác suất phổ biến để mô tả hiện tượng fading trong các hệ thống truyền thông quang học không dây. Ở phía nhận, Bob cũng sử dụng một bộ tạo ngẫu nhiên để lựa chọn cơ sở đo. Photon nhận được sẽ đi

qua bộ chia cơ sở và sau đó được đo trong một trong hai cơ sở (tuyến tính hoặc chéo). Nếu Bob chọn đúng cơ sở như Alice đã dùng, khả năng đo đúng bit là cao; ngược lại, nếu chọn sai cơ sở, kết quả đo là ngẫu nhiên với xác suất đúng 50%. Sau khi kết thúc quá trình truyền, Alice và Bob sử dụng một kênh công khai để trao đổi thông tin về cơ sở đo đã sử dụng (nhưng không tiết lộ giá trị bit), từ đó giữ lại các bit có cơ sở trùng khớp để hình thành khóa lượng tử chung.



Hình 16 - Sơ đồ của hệ thống với nhiễu

Nhằm đánh giá hiệu suất thực tế của giao thức phân phối khóa lượng tử BB84 trong điều kiện kênh truyền bị nhiễu, tôi tiến hành một mô phỏng mở rộng, trong đó quá trình đo tại Bob chịu ảnh hưởng của fading Gamma–Gamma – mô hình thường dùng để mô phỏng các kênh quang học tự do (FSO) trong điều kiện khí quyển không lý tưởng.

Trong mô hình này, Alice và Bob thực hiện giao thức BB84 truyền thống, tuy nhiên kết quả đo của Bob bị nhiễu xác suất do cường độ tín hiệu bị thay đổi ngẫu nhiên bởi fading. Hơn nữa, quá trình chuẩn bị trạng thái tại phía Alice cũng được thiết kế để có thể mô phỏng các sai lệch thực nghiệm bằng cách đưa vào xác suất áp dụng sai cổng Hadamard.

Mô phỏng được thực hiện với nhiều giá trị SIZE_TX khác nhau, mỗi giá trị tương ứng với số bit lượng tử được truyền. Đối với mỗi kích thước, mô phỏng lặp lại nhiều lần nhằm thu được kết quả trung bình về:

- Tỷ lệ lỗi bit lượng tử (QBER),
- Hiệu suất giữ khóa hợp lệ (Key Efficiency),
- Tỷ lệ giữ lại khóa so với số bit truyền.

Kết quả cho phép đánh giá mối quan hệ giữa kích thước đầu vào và các chỉ tiêu đánh giá hiệu suất. Dựa trên các ngưỡng yêu cầu về độ bảo mật và hiệu suất thực tế, mô phỏng cũng đề xuất một giá trị SIZE_TX tối ưu, phù hợp cho ứng dụng thực nghiệm hoặc triển khai thực tế.

Trong mô phỏng, fading *Gamma–Gamma* được áp dụng như sau:

Mỗi qubit khi truyền đến phía người nhận (Bob) sẽ trải qua một hệ số khuếch đại hoặc suy giảm cường độ ánh sáng, gọi là hệ số fading. Hệ số này được sinh ngẫu nhiên từ phân phối Gamma–Gamma với hai tham số đặc trưng: α và β, tương ứng với mức độ nhiễu loạn nhỏ và lớn. Sau đó, hệ số này được nhân với xác suất phát hiện trạng thái lượng tử của qubit. Việc này mô phỏng chính xác tác động của fading đến xác suất đo thành công qubit tại phía Bob: fading cao làm tăng khả năng đo chính xác, fading thấp có thể dẫn đến sai lệch trong quá trình đo, từ đó dẫn đến lỗi bit.

Việc mô phỏng theo cách này không can thiệp vào bản chất toán học của kênh lượng tử (tức là không sửa đổi trực tiếp trạng thái lượng tử), mà đưa fading vào thông qua quá trình đo, đúng như bản chất vật lý của hệ thống quang học chịu ảnh hưởng bởi nhiễu loạn.

QBER là tỷ lệ phần trăm bit mà người nhận (Bob) thu được khác với bit mà người gửi (Alice) gửi đi, tính trên tập hợp các bit kiểm tra (check bits) được công bố công khai. *QBER* phản ánh mức độ nhiễu trong kênh truyền và là chỉ tiêu quan trọng để đánh giá tính bảo mật của hệ thống. Một hệ thống được xem là an toàn nếu *QBER* nhỏ hơn ngưỡng xác định trước (thường là 11%).

Công thức tính *QBER*:

$$QBER = \frac{Số\ bit\ sai}{Tổng\ số\ bit\ kiểm\ tra} \times 100\% \tag{1.19}$$

Hiệu suất khóa lượng tử (*Key Efficiency*) phản ánh tỷ lệ bit có thể được giữ lại sau bước sàng lọc cơ sở đo (*basis reconciliation*), tức là sau khi Alice và Bob loại bỏ các bit mà họ đã sử dụng các cơ sở khác nhau trong quá trình mã hóa và đo. Đây là bước đặc trưng của giao thức BB84 nhằm đảm bảo rằng chỉ những bit thu được với xác suất cao nhất mới được giữ lại để tạo khóa.

Công thức tính hiệu suất khóa:

$$Hiệu\ suất\ khóa = \frac{Số\ bit\ trùng\ cơ\ sở}{Tổng\ số\ bit\ truyền} \times 100\% \tag{1.20}$$

Sau khi so sánh công khai một phần khóa để tính *QBER* (bằng cách sử dụng một nửa khóa làm check bits), chỉ phần còn lại của khóa thô được giữ lại để sử dụng trong quá trình trích xuất khóa an toàn. Tỷ lệ giữ lại khóa phản ánh tỷ lệ bit hữu dụng cuối cùng so với tổng số bit truyền. Chỉ số này thường nhỏ hơn hiệu suất khóa, do việc hi sinh một phần khóa thô để kiểm tra độ an toàn. Tỷ lệ giữ lại cao đồng nghĩa với việc ít bị mất mát dữ liệu trong quá trình xác minh khóa.

Công thức tính tỷ lệ giữ lại khóa:

$$Tỷ\ lệ\ giữ\ lại\ khóa = \frac{Số\ bit\ sau\ kiểm\ tra\ lỗi}{Tổng\ số\ bit\ truyền} \times 100\% \tag{1.21}$$

Số bit truyền tối ưu N^* trong mô phỏng BB84 dưới fading Gamma–Gamma được tính bằng cách cân bằng giữa ba yếu tố quan trọng: tỷ lệ lỗi bit lượng tử (*QBER*): càng thấp càng tốt (để đảm bảo bảo mật); hiệu suất giữ khóa hợp lệ (*Key Efficiency*): càng cao càng tốt (để tận dụng tối đa bit truyền); tỷ lệ giữ lại khóa (*Retained Key Ratio*): cho thấy phần khóa còn lại sau kiểm tra lỗi, càng cao càng hiệu quả.

Khi tăng số bit truyền *N*, các chỉ số trên thay đổi: QBER thường giảm do hiệu ứng trung bình, nhưng có thể sẽ tăng nếu fading mạnh. Hiệu suất giữ khóa ổn định nếu cơ sở chọn ngẫu nhiên đều. Tỷ lệ giữ lại khóa thường cao hơn khi QBER thấp (ít bit bị loại bỏ khi kiểm tra).

Để tối ưu, ta cần chọn *N* sao cho:

- $QBER(N) <$ ngưỡng an toàn (thường $là$ 11%).
- Hiệu suất khóa$(N) >$ ngưỡng hiệu suất tối thiểu (thường là 30%).
- Tỷ lệ giữ lại cao nhất trong số các *N* thỏa mãn hai điều kiện trên.

Để chọn ra N^*, mô phỏng sử dụng một hàm mục tiêu:

$$Score(N) = \frac{KeyEfficiency(N) \times \ Retained\ Key\ Ratio(N)}{1 + \ QBER(N)} \times 100\% \qquad (1.22)$$

Việc xây dựng một hàm mục tiêu phù hợp là bước then chốt trong quá trình tối ưu hóa tham số SIZE_TX – số lượng bit lượng tử được truyền trong mỗi phiên của giao thức BB84. Mục tiêu của hàm này là đánh giá toàn diện hiệu suất của hệ thống phân phối khóa lượng tử trong điều kiện có nhiễu fading Gamma–Gamma, từ đó lựa chọn giá trị SIZE_TX tối ưu nhất cho ứng dụng thực tế. Trong quá trình mô phỏng, ba chỉ số được theo dõi:

- QBER (Quantum Bit Error Rate) – đo mức độ sai lệch giữa bit gửi và bit nhận.
- Key Efficiency – tỷ lệ số bit trùng cơ sở (tức là có thể giữ lại để tạo khóa).
- Retained Key Ratio – tỷ lệ phần khóa còn lại sau kiểm tra lỗi (bit không bị loại bỏ sau bước tính QBER).

Mỗi chỉ số đều mang một vai trò riêng và phản ánh một khía cạnh hiệu suất hoặc bảo mật:

- QBER càng thấp → hệ thống càng an toàn trước sự nghe lén hoặc lỗi đo.
- Key Efficiency càng cao → quá trình truyền tải hiệu quả hơn, ít lãng phí.
- Retained Key Ratio càng cao → nhiều bit khóa còn lại thực sự có thể sử dụng sau quá trình xử lý.

Do đó, hàm mục tiêu cần tích hợp đầy đủ cả ba yếu tố. Nếu chỉ chọn một trong ba, ví dụ như tối ưu hóa QBER, thì có thể đánh đổi hiệu suất (ít bit giữ lại). Ngược lại, nếu chỉ tối ưu hóa hiệu suất mà bỏ qua QBER, hệ thống có thể không an toàn. Từng thành phần chi tiết như sau:

Tử số: Lấy tích của Key Efficiency và Retained Key Ratio nhằm phản ánh đồng thời hai yếu tố hiệu suất. Tích giữa hai chỉ số hiệu suất phản ánh mối quan hệ phụ thuộc: nếu một trong hai thấp, hiệu suất tổng thể cũng giảm mạnh. Ví dụ:

- Nếu Key Efficiency cao (nhiều bit trùng cơ sở), nhưng Retained Ratio thấp (nhiều bit sai do lỗi) thì hệ thống vẫn không hiệu quả.
- Nếu chỉ lấy trung bình hoặc tổng, hệ thống có thể được đánh giá cao dù một thành phần thực sự rất thấp.
- Giá trị lớn nhất đạt được khi cả hai chỉ số gần 1 (hoặc 100%) → khi đó hệ thống rất hiệu quả.
- Giá trị thấp nếu một trong hai gần 0 → hệ thống kém.

Tích tạo nên tính cảnh báo nghiêm khắc: bit hiệu quả phải vừa trùng cơ sở vừa vượt qua kiểm tra lỗi mới được tính là hữu dụng. Điều này phù hợp với bản chất hệ thống QKD – chỉ giữ lại các bit vừa đáng tin cậy về cấu trúc (cùng cơ sở) vừa an toàn (qua được kiểm tra QBER).

Mẫu số là $1 + QBER(N)$, nó phản ánh vai trò ràng buộc bảo mật, tức $QBER$ là thứ "kéo lùi" toàn bộ hệ thống nếu quá lớn. $QBER$ phản ánh mức độ nhiễu hoặc lỗi, nên đóng vai trò như một yếu tố trừng phạt (penalty term). Nếu mẫu số là $QBER$, thì với $QBER = 0$ sẽ xảy ra chia cho 0 → không xác định, nếu $QBER$ rất nhỏ, điểm số tăng đột ngột quá mức → không thực tế. Do đó, thêm 1 để tránh lỗi toán học và ổn định hóa biểu thức.

Dưới đây là phần so sánh trực quan giữa các hàm mục tiêu khác nhau khi cả 3 tham số Key Efficiency (KE), Retained Key Ratio (RKR) và QBER cùng biến thiên từ 0 đến 1. Ta sẽ xây dựng và vẽ 4 hàm mục tiêu phổ biến để minh họa:

Tên hàm	Biểu thức
Hàm 1 (gốc)	$\boxed{S_1 = \frac{KE \cdot RKR}{1 + QBER} \times 100}$
Hàm 2 (tổng – lỗi)	$S_2 = \frac{KE + RKR}{2} - QBER$
Hàm 3 (hiệu suất – lỗi)	$S_3 = KE \cdot RKR - QBER$
Hàm 4 (log phi tuyến)	$S_4 = \log\left(1 + \frac{KE \cdot KRR}{QBER + \epsilon}\right)$

Độ nhạy trung bình với QBER là đại lượng đo mức độ thay đổi trung bình của giá trị hàm mục tiêu (Score) khi QBER tăng lên một lượng nhỏ, cụ thể là 0.01 (tức 1%).

Giả sử có một hàm mục tiêu $Score = f(KE, RKR, QBER)$. Khi giữ KE và RKR cố định, nếu $QBER$ tăng từ $q \rightarrow q + 1{,}1$, thì Score thay đổi từ $f(q) \rightarrow f(q + 0{,}01)$
Độ nhạy gần đúng (approximate sensitivity) được tính như sau:

$$Sensitivity \approx \frac{|f(q) - f(q + 0{,}01)|}{0{,}01}$$

Sau đó, tính giá trị trung bình của tất cả các độ nhạy này trên toàn bộ tập dữ liệu (các cấu hình KE, RKR, QBER) → đó là độ nhạy trung bình.

Tương quan Spearman (Spearman's rank correlation coefficient) là một phép đo thống kê cho biết mức độ liên hệ đơn điệu giữa hai biến, dựa trên thứ hạng (rank) thay vì giá trị thực của dữ liệu. Khác với hệ số tương quan Pearson vốn đánh giá mối quan hệ tuyến tính, Spearman không yêu cầu điều kiện đó, nên rất phù hợp để phân tích các mối quan hệ phi tuyến hoặc có nhiễu.

Trong nghiên cứu này, tương quan Spearman được dùng để đánh giá mối liên hệ giữa QBER và Score – tức là kiểm tra xem khi QBER tăng (đồng nghĩa với nhiễu kênh tăng), thì Score có xu hướng tăng hay giảm, và mối liên hệ đó mạnh hay yếu.

Nếu Score là một hàm đánh giá tốt, ta mong đợi rằng khi QBER tăng, Score sẽ giảm – nghĩa là tương quan âm. Ngược lại, nếu Spearman gần 0 hoặc dương, điều đó cho thấy Score không phản ứng đúng với nhiễu, hoặc phản ứng sai hướng.

Bảng 8 - Bảng tham chiếu tương quan Spearman

Giá trị	Ý nghĩa
$+1.0$	Tương quan hoàn hảo thuận chiều
$\sim +0.5$	Tương quan thuận chiều trung bình
~ 0	Không tương quan
~ -0.5	Tương quan nghịch chiều trung bình
-1	Tương quan hoàn hảo nghịch chiều

Để đánh giá mức độ phù hợp của các hàm mục tiêu (Score1 đến Score4) trong việc phản ánh chất lượng của kênh lượng tử, ta phân tích dựa trên hai tiêu chí:

1. Tương quan Spearman với QBER – đo lường mức độ phụ thuộc đơn điệu giữa Score và QBER (QBER tăng thì Score nên giảm).
2. Độ nhạy trung bình khi QBER tăng 0.01 – đo lường mức thay đổi của Score khi QBER tăng nhẹ, thể hiện khả năng phản ứng của hàm.

Bảng 9 - Bảng so sánh tương quan và độ nhạy

Score	Spearman với QBER	Mức độ tương quan	Độ nhạy trung bình	Mức phản ứng
Score 1	-0.1600	Rất yếu (nghịch chiều)	0.1255	Rất thấp
Score 2	-0.8170	Rất mạnh (nghịch chiều)	1.0000	Cao và ổn định
Score 3	-0.7982	Rất mạnh (nghịch chiều)	1.0000	Cao và ổn định
Score 4	-0.5210	Trung bình (nghịch chiều)	3.5069	Rất nhạy

Bảng 9 là kết quả thống kê tương quan và độ nhạy của bốn hàm mục tiêu, Score 2 và Score 3 có hệ số tương quan Spearman rất mạnh âm (≈ -0.8), chứng tỏ chúng phản ánh tốt QBER: khi QBER tăng thì Score giảm rõ rệt. Đồng thời, độ nhạy đều bằng 1.0000, cho thấy phản ứng ổn định và có khả năng phân biệt các mức độ lỗi. Score 4 tuy có tương quan thấp hơn (ρ = -0.5210) nhưng độ nhạy rất cao (3.51), nghĩa là Score 4 thay đổi rất mạnh khi QBER tăng. Tuy nhiên, độ nhạy quá cao có thể khiến Score dao động mạnh, dễ bị ảnh hưởng bởi nhiễu (noise), không ổn định trong thực tế. Score 1 thể hiện kém nhất: tương quan rất yếu với *QBER* (ρ = -0.16) và độ nhạy thấp (0.1255) → không phù hợp làm chỉ số phản ánh chất lượng kênh.

Các tham số mô phỏng được thể hiện trong *Bảng 10* như sau:

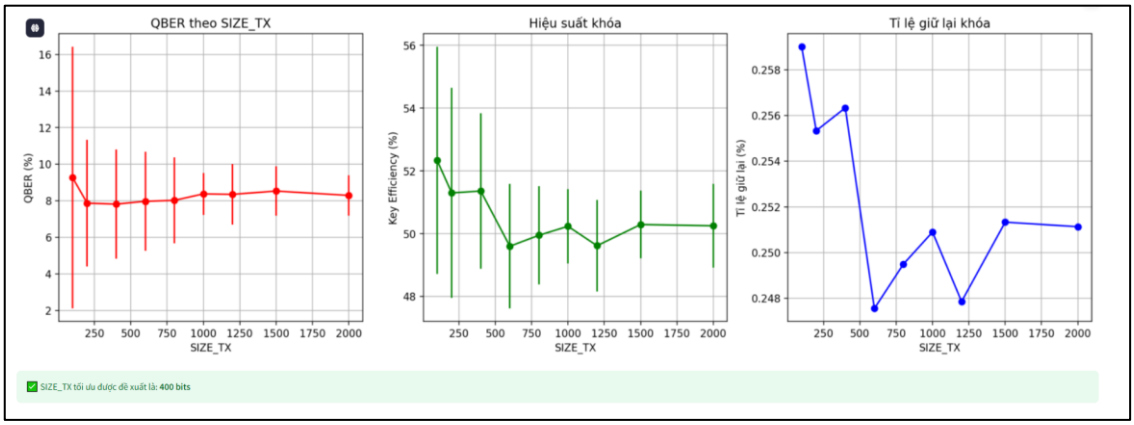
Bảng 10 - Các tham số mô phỏng

Tham số	Giá trị	Mô tả
size	[100,…2000]	Các giá trị số bit truyền (SIZE_TX) được thử nghiệm
num_trials	30	Số lần lặp lại mỗi mô phỏng để lấy trung bình
alpha	3.0	Tham số hình dạng (shape) trong mô hình fading
beta	2.5	Tham số tỷ lệ (scale) trong mô hình fading

epsilon	1e-6	Tránh chia cho 0 trong các công thức tính toán
qber_thresh	11	Ngưỡng QBER tối đa chấp nhận để chọn SIZE_TX tối ưu
min_eff	30	Ngưỡng hiệu suất khóa tối thiểu để chấp nhận SIZE_TX

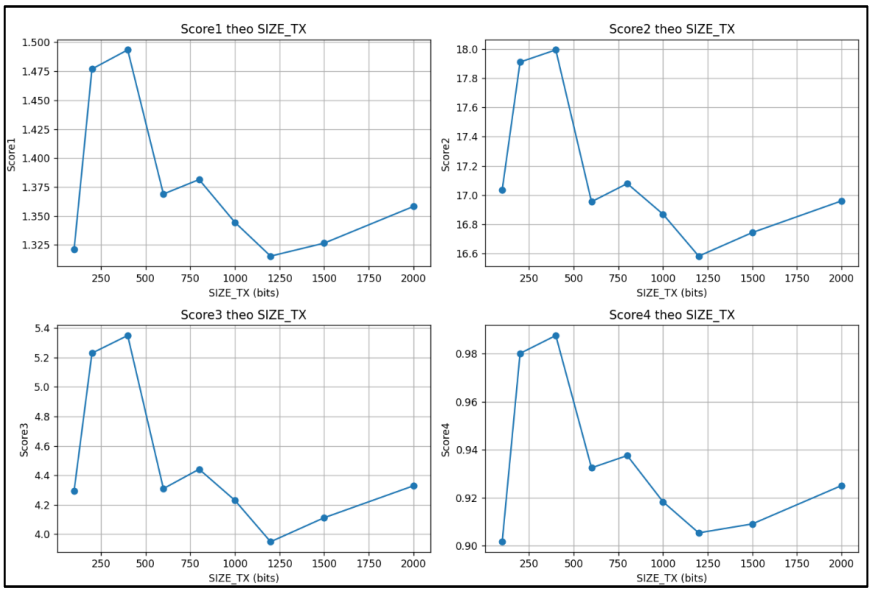
Hình 17 là kết quả mô phỏng, cho thấy tỷ lệ lỗi bit lượng tử (*QBER*) có xu hướng giảm dần khi kích thước truyền (*SIZE_TX*) tăng từ 100 đến khoảng 400 bits. Tại *SIZE_TX* nhỏ, QBER dao động mạnh với sai số lớn do ảnh hưởng đáng kể của nhiễu kênh Gamma-Gamma, trong khi khi *SIZE_TX* tăng, *QBER* dần ổn định quanh mức *8%*. Điều này phản ánh vai trò của kích thước truyền trong việc làm trung bình hóa tác động của fading ngẫu nhiên, từ đó cải thiện độ chính xác trong truyền dẫn khóa lượng tử.

Hiệu suất giữ khóa (*Key Efficiency*) trong mô phỏng dao động quanh ngưỡng *50–52%*, phù hợp với kỳ vọng lý thuyết của giao thức BB84, khi xác suất chọn đúng cơ sở là *50%*. Sai số của hiệu suất cao hơn ở *SIZE_TX* nhỏ do dữ liệu ít dễ bị nhiễu ảnh hưởng, trong khi ở *SIZE_TX* lớn hơn, hiệu suất trở nên ổn định. Điều này cho thấy dù kênh truyền có chịu ảnh hưởng bởi fading *Gamma-Gamma*, giao thức vẫn duy trì được hiệu quả trong việc phân phối khóa với xác suất đúng cơ sở gần như tối ưu.



Hình 17 - Kết quả mô phỏng BB84 dưới ảnh hưởng của fading

Tỷ lệ giữ lại khóa (*Retained Key Ratio*) đạt mức cao nhất tại *SIZE_TX = 400* bits *(~25.8%)* và ổn định quanh mức *25%* tại các kích thước khác. Điều này chứng tỏ rằng *SIZE_TX = 400* là điểm tối ưu: vừa đủ lớn để trung bình hóa nhiễu, vừa không quá lớn để gây lãng phí tài nguyên tính toán. Do đó, kết quả mô phỏng đề xuất kích thước truyền tối ưu là *400 bits*, giúp cân bằng tốt giữa độ chính xác, hiệu suất và khả năng giữ lại khóa lượng tử sau xử lý lỗi.

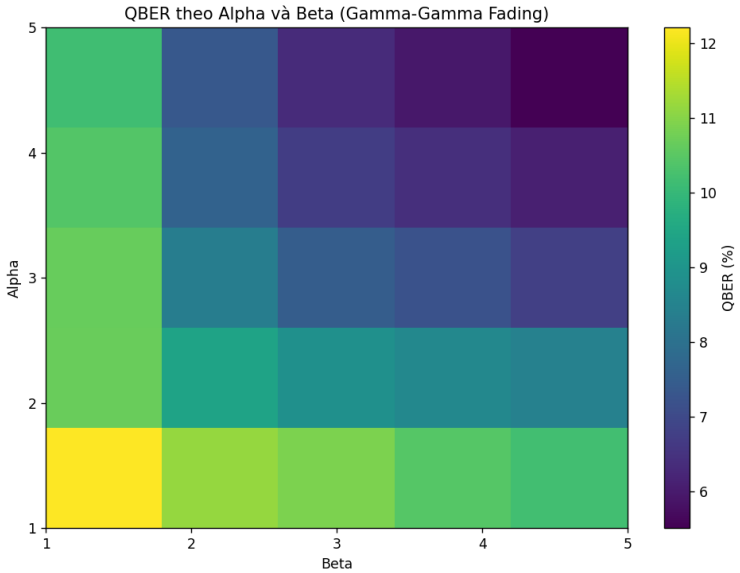


Hình 18 - Kết quả Score của bốn hàm mục tiêu

Hình 18 là kết quả mô phỏng *Score* theo kích thước bit của bốn hàm mục tiêu. Từ đồ thị, ta có thể thấy cả bốn hàm đều cho kết quả tối ưu tại 400 bit, hơn nữa hình dáng của đồ thị cũng khá tương đồng, cho thấy bốn hàm đều có thể sử dụng làm hàm tối ưu trong kịch bản mô phỏng này. Ta có thể kết luận số bit tối ưu dùng để truyền là 400 bit.

3.5 Mô phỏng đánh giá các tham số hình dạng của fading Gamma-Gamma

Với mục tiêu mô phỏng sự thay đổi của tỷ lệ lỗi bit lượng tử (*QBER*) dưới ảnh hưởng của fading Gamma-Gamma, thông qua việc thay đổi các tham số hình dạng. Các kết quả được tổng hợp và hiển thị bằng biểu đồ nhiệt (heatmap), giúp đánh giá mức độ suy hao kênh ảnh hưởng đến độ chính xác truyền khóa lượng tử.



Hình 19 - Kết quả mô phỏng QBER theo alpha và beta

Với *SIZE_TX = 400*, ta đang dùng kích thước truyền đã được xác định là tối ưu trong mô phỏng trước đó, từ đó giúp đảm bảo kết quả QBER thu được là đáng tin cậy. Mô phỏng này giúp định lượng tác động của các mức fading khác nhau đến độ chính xác truyền khóa lượng tử.

Biểu đồ nhiệt cho thấy rõ ràng xu hướng giảm dần của tỷ lệ lỗi bit lượng tử (*QBER*) khi cả hai tham số alpha và beta tăng. Điều này phản ánh trực tiếp bản chất vật lý của mô hình *Gamma-Gamma*: alpha và beta càng lớn thì biến động cường độ tín hiệu càng nhỏ, tức là nhiễu do hiện tượng nhiễu xạ và dao động cường độ do truyền qua khí quyển càng yếu. Do đó, khi kênh truyền ít bị méo, hệ thống QKD thu được kết quả chính xác hơn và *QBER* thấp hơn.

Các vùng phía dưới bên trái của biểu đồ – nơi alpha và beta đều nhỏ (gần giá trị *1*) – có màu sáng, tức *QBER* cao (trên *10–12%*). Điều này tương ứng với điều kiện truyền khắc nghiệt hơn, khi fading Gamma-Gamma mô tả nhiễu loạn mạnh trong môi trường truyền ánh sáng tự do (FSO), dẫn đến suy giảm chất lượng truyền khóa lượng tử nghiêm trọng.

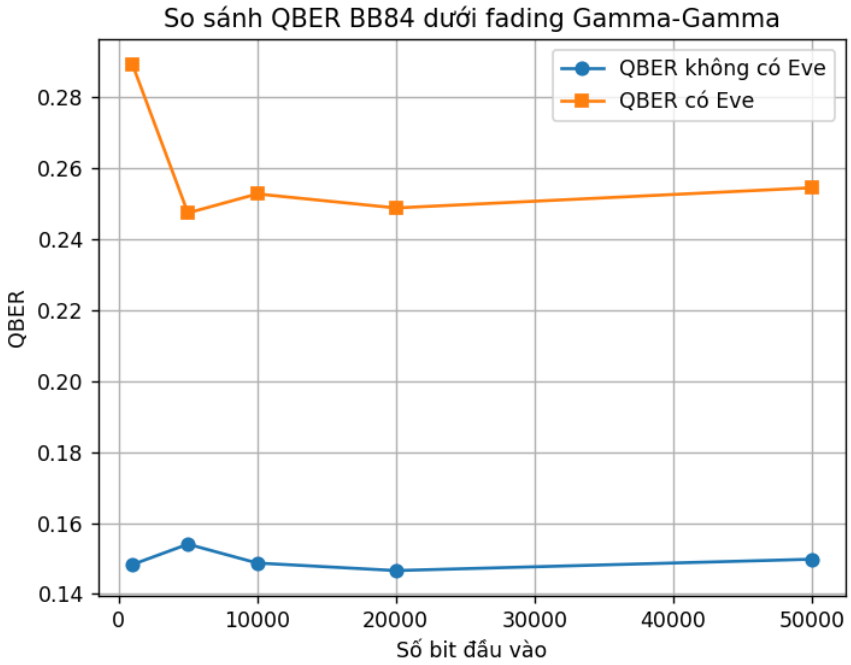
Ở góc trên phải của biểu đồ (*alpha ≈ 5, beta ≈ 5*), *QBER* chỉ còn khoảng *5.5–6%,* là mức rất tốt trong hệ thống QKD. Điều này cho thấy nếu đảm bảo điều kiện truyền ổn định hơn (ví dụ bằng việc dùng chuẩn bị chùm tia tốt, giảm nhiễu môi trường hoặc truyền trong thời điểm lý tưởng), thì hiệu suất bảo mật và độ chính xác của khóa lượng tử có thể được cải thiện đáng kể.

3.6 Mô phỏng đánh giá ảnh hưởng của Eve với fading Gamma-Gamma

Thực hiện mô phỏng giao thức phân phối khóa lượng tử BB84 trong hai kịch bản: có và không có kẻ nghe lén (Eve). Trong cả hai trường hợp, Alice tạo ra một chuỗi bit ngẫu nhiên cùng với cơ sở đo tương ứng, sau đó gửi các bit qua kênh truyền chịu ảnh hưởng của fading Gamma-Gamma, mô phỏng sự thay đổi cường độ tín hiệu do môi trường.

Ở kịch bản không có Eve, Bob chọn cơ sở đo ngẫu nhiên và nhận tín hiệu từ Alice; nếu cơ sở trùng khớp, Bob có thể nhận đúng hoặc sai bit do nhiễu từ kênh fading.

Ở kịch bản có Eve, Eve thực hiện tấn công kiểu "intercept-resend" bằng cách đo các qubit từ Alice với cơ sở ngẫu nhiên rồi gửi lại cho Bob. Quá trình đo và gửi lại này gây ra sai lệch, làm tăng xác suất lỗi. Sau khi truyền xong, Alice và Bob so sánh cơ sở đo, giữ lại những bit trùng cơ sở và tính toán QBER (Quantum Bit Error Rate) để đánh giá mức độ sai lệch giữa bit của hai bên. Mô phỏng được lặp lại với nhiều độ dài chuỗi bit khác nhau để quan sát sự thay đổi của QBER trong từng kịch bản.



Hình 20 - Kết quả mô phỏng ảnh hưởng của Eve

Hình 18 mô tả sự so sánh xác suất lỗi lượng tử (*QBER*) của giao thức BB84 dưới ảnh hưởng của fading Gamma-Gamma trong hai trường hợp: có và không có kẻ tấn công Eve. Có thể rút ra các nhận xét sau:

QBER cao hơn khi có sự hiện diện của Eve: Trong tất cả các mức số bit đầu vào, *QBER* trong trường hợp có Eve luôn cao hơn đáng kể so với trường hợp không có Eve. Điều này cho thấy tấn công kiểu "*intercept-resend*" gây ra nhiễu loạn đáng kể trong quá trình phân phối khóa, làm tăng tỷ lệ lỗi.

QBER ổn định khi số bit tăng: Cả hai đường *QBER* đều có xu hướng ổn định hơn khi số bit đầu vào tăng lên. Điều này phản ánh rằng khi khối lượng dữ liệu đủ lớn, ảnh hưởng của nhiễu ngẫu nhiên và biến động trong mô phỏng giảm đi, giúp kết quả *QBER* trở nên ổn định hơn.

QBER trung bình thấp hơn trong kịch bản không có tấn công: Ở kịch bản không có Eve, *QBER* dao động xung quanh giá trị *~0.145–0.155*, cho thấy ảnh hưởng chủ yếu là từ fading Gamma-Gamma của kênh truyền. Trong khi đó, *QBER* ở kịch bản có Eve dao động quanh mức *~0.25–0.29*, thể hiện tác động kết hợp của cả fading và sự đo sai cơ sở từ phía Eve.

Nhìn chung, kết quả này minh chứng rằng *QBER* có thể là chỉ số hiệu quả để phát hiện sự hiện diện của kẻ nghe lén trong hệ thống BB84, đặc biệt là trong điều kiện fading phức tạp như Gamma-Gamma.

3.7 Kết luận chương

Chương này đã mô phỏng thành công quá trình truyền khóa lượng tử trong các điều kiện lý tưởng và thực tế. Kết quả cho thấy fading Gamma-Gamma ảnh hưởng lớn đến QBER, và Eve có thể bị phát hiện nhờ chỉ số lỗi. Các kết quả mô phỏng khẳng định tính hiệu quả và độ nhạy của giao thức BB84 trước các yếu tố gây nhiễu và nghe lén.

KẾT LUẬN

Trong đồ án này, chúng ta đã tiến hành một nghiên cứu chuyên sâu và toàn diện về giao thức phân phối khóa lượng tử BB84, được triển khai trong môi trường truyền thông sử dụng ánh sáng khả kiến (Visible Light Communication - VLC). Đặc biệt, quá trình mô phỏng và phân tích được thực hiện dưới ảnh hưởng của mô hình fading Gamma-Gamma, một mô hình kênh thực tế rất phổ biến và phù hợp để mô tả sự biến đổi tín hiệu trong các hệ thống quang học trong điều kiện môi trường phức tạp.

Ban đầu, đồ án trình bày một tổng quan chi tiết về các khái niệm cơ bản liên quan đến phân phối khóa lượng tử (Quantum Key Distribution - QKD), nguyên lý hoạt động của giao thức BB84, cũng như đặc điểm kỹ thuật và ưu nhược điểm của kênh truyền VLC trong bối cảnh ứng dụng thực tế. Ngoài ra, chúng ta cũng đã làm rõ bản chất của mô hình fading Gamma-Gamma, vốn là một trong những yếu tố gây ra hiện tượng suy giảm tín hiệu và làm tăng tỷ lệ lỗi trong quá trình truyền thông lượng tử qua kênh quang học.

Tiếp theo, đồ án xây dựng và phát triển mô hình toán học chính xác cho hệ thống truyền thông VLC kết hợp với giao thức BB84. Mô hình này bao gồm các thành phần quan trọng như đặc tính phát xạ ánh sáng của LED, quá trình điều chế và phát sóng các bit lượng tử, cùng với ảnh hưởng của fading Gamma-Gamma lên tín hiệu nhận được tại phía người nhận (Bob).

Phân tích sâu hơn về tác động của fading Gamma-Gamma được thực hiện thông qua việc đánh giá các thông số hiệu suất của hệ thống, điển hình là tỷ lệ lỗi bit lượng tử (Quantum Bit Error Rate - QBER) và hiệu suất sinh khóa (Key Efficiency). Những phân tích này giúp làm rõ mức độ ảnh hưởng của các yếu tố môi trường và nhiễu lên chất lượng và tính bảo mật của quá trình phân phối khóa lượng tử.

Đồng thời, đồ án cũng mô phỏng toàn bộ quá trình thực thi giao thức BB84, trong đó đặc biệt chú trọng đến việc kiểm tra vai trò của nhiễu fading cũng như sự hiện diện của kẻ nghe lén (Eve). Qua đó, chúng ta nhận thấy rằng sự xuất hiện của Eve làm tăng đáng kể giá trị QBER, từ đó cung cấp cơ sở để phát hiện và ngăn chặn các cuộc tấn công tiềm năng, bảo đảm an toàn cho hệ thống phân phối khóa.

Ngoài ra, dựa trên kết quả mô phỏng và phân tích, đồ án đề xuất một kích thước truyền khóa tối ưu, cân nhắc kỹ lưỡng giữa các yếu tố như hiệu suất sinh khóa, mức độ an toàn cần thiết, và khả năng chống lại các cuộc tấn công nghe lén. Điều này góp phần định hướng cho việc thiết kế các hệ thống QKD thực tế với hiệu quả và độ tin cậy cao.

Kết quả nghiên cứu và mô phỏng đã chứng minh rằng fading Gamma-Gamma có ảnh hưởng rõ rệt đến chất lượng và hiệu quả của quá trình phân phối khóa lượng tử. Điều này đặt ra yêu cầu phải áp dụng các phương pháp hiệu chỉnh và điều chỉnh thích nghi nhằm duy trì và nâng cao tính bảo mật của hệ thống trong các điều kiện môi trường đa dạng và phức tạp. Đồng thời, việc sử dụng QBER làm thước đo phát hiện sự xuất hiện của kẻ nghe lén cũng được khẳng định là một công

cụ mạnh mẽ và hiệu quả trong việc giám sát an ninh của hệ thống phân phối khóa lượng tử.

Mặc dù đồ án đã đạt được những kết quả bước đầu quan trọng trong việc mô phỏng và phân tích giao thức phân phối khóa lượng tử BB84 trên kênh truyền VLC chịu ảnh hưởng của fading Gamma-Gamma, vẫn còn rất nhiều hướng phát triển tiềm năng có thể tiếp tục được khai thác nhằm nâng cao hiệu suất, độ tin cậy và tính ứng dụng thực tế của hệ thống.

Trước hết, một hướng nghiên cứu đáng chú ý là mở rộng mô hình kênh truyền để bao gồm các yếu tố môi trường phức tạp hơn như nhiễu đa đường, sự thay đổi nhiệt độ, và các hiện tượng vật lý phi tuyến trong kênh quang học. Việc phát triển mô hình kênh đa chiều, đa trạng thái sẽ giúp tăng độ chính xác của các kết quả mô phỏng và làm rõ thêm các thách thức thực tiễn mà hệ thống QKD phải đối mặt khi triển khai trong điều kiện ngoài phòng thí nghiệm.

Bên cạnh đó, việc kết hợp các kỹ thuật điều chế và mã hóa tiên tiến hơn, ví dụ như mã lượng tử sửa lỗi (Quantum Error Correction Codes) hoặc kỹ thuật phân phối khóa lượng tử dựa trên nhiều photon (multi-photon QKD), cũng là hướng đi quan trọng nhằm cải thiện khả năng chống chịu lỗi và tăng cường độ bảo mật cho hệ thống. Điều này đặc biệt cần thiết trong các môi trường truyền thông không ổn định hoặc có mức độ nhiễu cao.

Ngoài ra, tích hợp các thuật toán tối ưu vị trí và định hướng của nguồn phát ánh sáng cũng như bộ thu trong hệ thống VLC có thể góp phần giảm thiểu ảnh hưởng của fading Gamma-Gamma, từ đó nâng cao hiệu suất truyền khóa và giảm tỷ lệ lỗi. Sự kết hợp này mở ra khả năng ứng dụng hệ thống QKD trong các mạng truyền thông quang học thực tế, như hệ thống mạng trong nhà thông minh, giao tiếp trong môi trường công nghiệp hay các mạng viễn thông bảo mật cao.

Hơn nữa, nghiên cứu về các phương pháp phát hiện và ngăn chặn tấn công nghe lén phức tạp hơn, chẳng hạn như tấn công giả mạo photon (photon spoofing) hay tấn công trung gian (man-in-the-middle) trong môi trường VLC, cũng là vấn đề cấp thiết nhằm đảm bảo tính toàn vẹn và bảo mật cho hệ thống QKD trong tương lai.

Cuối cùng, việc xây dựng các mô phỏng thực nghiệm và phát triển phần cứng prototype cho hệ thống truyền thông QKD dựa trên VLC sẽ là bước tiến quan trọng để đánh giá hiệu quả và khả năng ứng dụng trong thực tế, đồng thời tạo tiền đề cho việc thương mại hóa công nghệ phân phối khóa lượng tử trong các hệ thống mạng hiện đại.

Tóm lại, với những nền tảng lý thuyết và kết quả mô phỏng đã đạt được, đồ án mở ra nhiều hướng nghiên cứu và ứng dụng đa dạng, góp phần thúc đẩy sự phát triển của công nghệ truyền thông lượng tử bảo mật trong kỷ nguyên số.

TÀI LIỆU THAM KHẢO

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India, 1984, pp. 175–179.

[2] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 145–195, Jan. 2002.

[3] D. Tsonev, S. Sinanovic, and H. Haas, "Complete modeling of nonlinear distortion in VLC systems," *Journal of Lightwave Technology*, vol. 34, no. 16, pp. 3822–3830, Aug. 2016.

[4] J. Armstrong, "OFDM for optical communications," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 189–204, Feb. 2009.

[5] M. Uysal, J. Li, and M. Kavehrad, "Error rate performance analysis of coded free-space optical links over Gamma-Gamma atmospheric turbulence channels," *IEEE Transactions on Wireless Communications*, vol. 5, no. 6, pp. 1229–1233, Jun. 2006.

[6] L. Andrews and R. Phillips, *Laser Beam Propagation through Random Media*, 2nd ed. SPIE Press, 2005.

[7] H. Elgala, R. Mesleh, and H. Haas, "Indoor optical wireless communication: Potential and state-of-the-art," *IEEE Communications Magazine*, vol. 49, no. 9, pp. 56–62, Sep. 2011.

[8] Z. Chen, L. Yin, and H. Chen, "Analysis of Quantum Key Distribution over Free-Space Optical Channels with Atmospheric Turbulence," *IEEE Photonics Journal*, vol. 12, no. 1, pp. 1–12, Feb. 2020.

[9] J. M. Kahn and J. R. Barry, "Wireless infrared communications," *Proceedings of the IEEE*, vol. 85, no. 2, pp. 265–298, Feb. 1997.

PHỤ LỤC

A. Mô phỏng đồ họa giải thuật BB84 có và không có Eve

a. Tổng quan những chức năng chính

Thành phần	Mô tả
`setup()`	Tạo canvas và khởi tạo hệ thống
`init()`	Khởi tạo các bit, cơ sở, qubit cho Alice, Eve và Bob. Tính khóa và QBER
`draw()`	Vẽ mô phỏng trực quan quá trình truyền qubit
`drawState()`	Vẽ trạng thái qubit tại các điểm xác định
`m() và cm()`	Đo qubit trong một cơ sở và chuyển nó thành bit

b. Chi tiết chức năng BB84

```
/*****************************************************
****
 * MÔ PHỎNG BB84 (phiên bản rút gọn, không có Eve) bằng
p5.js
 * ---------------------------------------------------
----
 * • Alice tạo n qubit theo cơ sở ngẫu nhiên (Z hoặc X).
 * • Bob đo các qubit bằng cơ sở ngẫu nhiên của riêng
mình.
 * • Hai bên so sánh cơ sở: chỉ những vị trí trùng cơ sở
(dd[i]=true)
 *   mới được giữ lại để hình thành khóa lượng tử.
 * • Chuột trái  : chạy lại mô phỏng (gọi init()).
 * • Chuột phải  : chuyển đổi cách hiển thị (ký hiệu
Dirac ⇔ biểu tượng).

*****************************************************
***/


/* ---------- BIẾN TOÀN CỤC ---------- */
```

```
let n        = 16;      // Số qubit trong mỗi lần chạy mô phỏng
let dirac    = false;   // true  → hiển thị ký hiệu Dirac |0⟩,|1⟩,|+⟩,|−⟩
                        // false → hiển thị bằng biểu tượng hình học
let alice    = [[],[],[]]; // [bit gốc, cơ sở mã hóa, trạng thái qubit]
let bob      = [[],[],[]]; // [bit đo, cơ sở đo, trạng thái sau đo]
let dd       = [];      // dd[i] = true nếu Alice & Bob dùng cùng cơ sở ở vị trí i
let keyAlice = "";      // Chuỗi khóa sau sàng lọc (từ phía Alice)
let keyBob   = "";      // Chuỗi khóa sau sàng lọc (từ phía Bob)

/* ---------- p5.js: HÀM KHỞI TẠO ---------- */
function setup() {
  createCanvas(650, 450); // Canvas 650×450 px
  init();                 // Tạo dữ liệu qubit & cấu hình ban đầu
}

/* ---------- HÀM TẠO DỮ LIỆU MỚI ---------- */
function init() {
  /* Cấu hình vẽ chữ & đường kẻ */
  textAlign(CENTER, CENTER); // Căn chữ giữa ô
  strokeWeight(2);           // Độ dày nét vẽ

  /* Đặt lại tất cả các biến toàn cục */
  n        = 16;
```

```
  dirac    = false;
  alice    = [[], [], []];
  bob      = [[], [], []];
  dd       = [];
  keyAlice = "";
  keyBob   = "";

  /* ----- VÒNG LẶP TẠO & ĐO n QUBIT ----- */
  for (let i = 0; i < n; i++) {

    /* 1) Alice sinh bit gốc (0/1) ngẫu nhiên */
    alice[0].push(random([0, 1]));

    /* 2) Alice sinh cơ sở mã hóa ngẫu nhiên  (Z-basis
hoặc  X-basis)   *
     *    - Z-basis:   [[1,0],[0,1]]   tương đương |0⟩,|1⟩
*
     *    - X-basis:   [[√2/2,√2/2],[√2/2,-√2/2]]   tương
đương |+⟩,|−⟩    */
    alice[1].push(random([
      [[1, 0], [0, 1]],                                //
Z-basis
      [["√2/2", "√2/2"], ["√2/2", "-√2/2"]]          //
X-basis
    ]));

    /* 3) Bob chọn cơ sở đo ngẫu nhiên (cùng cấu trúc như
trên) */
    bob[1].push(random([
      [[1, 0], [0, 1]],
      [["√2/2", "√2/2"], ["√2/2", "-√2/2"]]
```

```
    ]));


    /* 4) Alice lấy trạng thái qubit phù hợp với bit & cơ
sở đã chọn
     *    alice[2][i] = hàng 0 hoặc 1 của ma trận cơ sở
(bit quyết định hàng)   */
    alice[2].push(alice[1][i][alice[0][i]]);


    /* 5) Bob đo qubit của Alice trong cơ sở Bob:    *
     *    - m()   : trả về trạng thái sau đo (collapse)
*
     *    - cm()  : chuyển trạng thái sau đo thành bit (0
hoặc 1)     */
    bob[2].push(m(alice[2][i], bob[1][i]));       //
Trạng thái sau đo
    bob[0].push(cm(bob[2][i], bob[1][i]));        // Bit
đo được


    /* 6) Kiểm tra hai cơ sở có trùng nhau?  -=> đánh dấu
vào dd[i]   *
     *    So sánh phần tử [0][0] ( =1 đối với Z-basis, ≠1
đối với X-basis)   */
    dd.push(alice[1][i][0][0] === bob[1][i][0][0]);


    /* 7) Nếu trùng cơ sở ⇒ giữ lại bit vào khóa lượng tử
*/
    if (dd[i]) {
      keyAlice += alice[0][i]; // Bit gốc của Alice
      keyBob   += bob[0][i];   // Bit Bob đo được
    }
  }
}
```

```
/* ---------- p5.js: VÒNG LẶP VẼ LIÊN TỤC ---------- */
function draw() {
  background(0);        // Xóa màn hình (màu đen)
  textSize(16);
  fill(255);            // Màu chữ trắng

  /* Hai chế độ hiển thị: Dirac hoặc biểu tượng *
   * dirac = true  → hiện chuỗi ký hiệu & ma trận *
   * dirac = false → vẽ hình (đường thẳng/chéo) tượng trưng cho cơ sở/qubit */
  if (dirac) {

    /* -------- CHẾ ĐỘ KÝ HIỆU DIRAC -------- */
    stroke(0);
    for (let i = 0; i < n; i++) {
      let y = (i + 0.75) * (375 / n);          // Vị trí dọc của hàng i

      /* Chỉ hiển thị khi: chuột nằm bên trái vùng key (mouseX<525)
       * hoặc vị trí i là trùng cơ sở (dd[i]=true) */
      if (mouseX < 525 || dd[i]) {
        /* Cột 1: biểu tượng truyền qubit */
        text("()=>", 125, y);

        /* Cột 2 & 3: hiển thị cơ sở (ma trận) của Alice và Bob (size 8) */
        textSize(8);
        text("[" + alice[1][i] + "]", 225, y); // Basis Alice
```

```
        text("[" + bob[1][i]   + "]", 325, y); // Basis
Bob
        textSize(16);

        /* Cột 4: kênh truyền  –  Cột 5: vùng khóa */
        text("[<>]", 425, y);
        text("Key",   525, y);

        /* ------ Tương tác khi rê chuột qua từng cột ---
--- */
        fill("red");
        if (mouseX < 125) {                          // Cột
truyền
          text("?", mouseX, y);
        } else if (mouseX < 225) {                   // Cột
bit Alice
          text(alice[0][i], mouseX, y);
        } else if (mouseX < 325) {                   // Cột
qubit (Alice)
          textSize(12);
          text("[" + alice[2][i] + "]", mouseX, y);
          textSize(16);
        } else if (mouseX < 425) {                   // Cột
qubit (Bob)
          textSize(12);
          text("[" + bob[2][i] + "]", mouseX, y);
          textSize(16);
        } else {                                     // Cột
bit Bob + bit Alice gốc
          text(bob[0][i], mouseX, y);                // Bit
Bob
```

```
          text(alice[0][i], 75, y);                // Bit Alice
        }
        fill(255);
      }
    }

  } else {

    /* -------- CHẾ ĐỘ BIỂU TƯỢNG ĐỒ HỌA -------- */
    for (let i = 0; i < n; i++) {
      let y = (i + 0.75) * (375 / n);

      if (mouseX < 525 || dd[i]) {
        /* Biểu tượng tam giác hồng: qubit truyền đi */
        noFill();
        stroke("pink");
        triangle(135, y, 115, y + 10, 115, y - 10);

        /* ---- Cột cơ sở Alice ---- */
        if (alice[1][i][0][0] === 1) {             // Z-basis
          stroke("red");
          line(225, y - 10, 225, y + 10);          // Đường đứng
          line(215, y, 235, y);                    // Đường ngang
        } else {                                   // X-basis
          stroke("skyblue");
```

```
          line(217.929, y - 7.071, 232.071, y + 7.071); // Chéo /
          line(217.929, y + 7.071, 232.071, y - 7.071); // Chéo \
        }
        rect(215, y - 10, 20, 20);  // Khung vuông bao cơ sở Alice

        /* ---- Cột cơ sở Bob ---- */
        if (bob[1][i][0][0] === 1) {               // Z-basis
          stroke("red");
          line(325, y - 10, 325, y + 10);
          line(315, y, 335, y);
        } else {                                   // X-basis
          stroke("skyblue");
          line(317.929, y - 7.071, 332.071, y + 7.071);
          line(317.929, y + 7.071, 332.071, y - 7.071);
        }
        rect(315, y - 10, 20, 20);  // Khung vuông bao cơ sở Bob

        /* Kênh truyền (mũi tên) & biểu tượng khóa */
        stroke("pink");
        quad(435, y, 425, y + 10, 415, y, 425, y - 10);
        stroke("yellow");
        ellipse(525, y, 15, 15);    // Khóa vàng
        line(533, y, 550, y);       // Răng khóa
        line(545, y - 5, 545, y);
        line(550, y - 5, 550, y);
```

```
        /* ------ Hiển thị trạng thái khi rê chuột qua ------ */
        push();
        if (mouseX < 125) {
          /* Cột truyền – không hiển thị gì thêm */
        } else if (mouseX < 225) {                  // Bit gốc Alice
          push();
          strokeWeight(0);
          fill(255);
          text(alice[0][i], mouseX, y);
          pop();
        } else if (mouseX < 325) {                  // Trạng thái qubit (Alice)
          drawState(mouseX, y, alice[2][i]);
        } else if (mouseX < 425) {                  // Trạng thái qubit (Bob)
          drawState(mouseX, y, bob[2][i]);
        } else {                                    // Bit Bob + bit Alice
          push();
          strokeWeight(0);
          fill(255);
          text(bob[0][i], mouseX, y);
          text(alice[0][i], 75, y);
          pop();
        }

        /* Chấm nhỏ bám theo chuột để người xem dễ theo dõi */
```

```
        if (mouseX < 525) ellipse(mouseX, y, 20);
        pop();
      }
    }
  }

  /* ----- Hiển thị khóa (nếu chuột bên phải) ----- */
  if (mouseX > 525) {
    push();
    strokeWeight(0);
    fill(255);
    text("Alice's Key: " + keyAlice, 175, 425);
    text("Bob's Key: "   + keyBob,   475, 425);
    pop();
  }
}

/* ------------------------------------------------
 *  HÀM m(state, basis)
 *  Mô phỏng phép đo qubit  →  trả về trạng thái hậu đo
 *  - Nếu đo đúng cơ sở  → trạng thái giữ nguyên.
 *  - Nếu đo sai cơ sở   → collapse ngẫu nhiên sang 1
 *    trong 2 vectơ của cơ sở đo.
 * ------------------------------------------------ */
function m(state, basis) {
  let a = JSON.stringify(state);
  let b = JSON.stringify(basis);
  return (b.indexOf(a) !== -1) ? state : random(basis);
}
```

```
/* ------------------------------------------------
 *  HÀM cm(state, basis)
 *  Chuyển trạng thái (vector) thành bit (0/1).
 *  Quy ước:  vector đầu tiên của basis → bit 0,
 *             vector thứ hai            → bit 1.
 * ------------------------------------------------ */
function cm(state, basis) {
  let a = JSON.stringify(state);
  let b = JSON.stringify(basis);
  if (b.indexOf(a) > 5) return 1;      // Khớp vector thứ 2
  if (b.indexOf(a) > 0) return 0;      // Khớp vector thứ 1
  return random([0, 1]);               // Khác cơ sở → ngẫu nhiên
}

/* ---------- XỬ LÝ SỰ KIỆN CHUỘT ---------- */
function mousePressed() {
  if (mouseButton === LEFT) {
    init();          // Chuột trái  → chạy lại mô phỏng
  } else {
    dirac = !dirac;  // Chuột phải → chuyển chế độ hiển thị
  }
}

/* ---------- HÀM VẼ TRẠNG THÁI QUBIT (biểu tượng) ---------- */
function drawState(x, y, state) {
```

```
  if (state[0] === 1) {                // |1⟩ – đường đứng đỏ
    stroke("red");       line(x, y - 10, x, y + 10);
  } else if (state[0] === 0) {         // |0⟩ – đường ngang đỏ
    stroke("red");       line(x - 10, y, x + 10, y);
  } else if (state[1] === "√2/2") {    // |+⟩ – đường chéo /
    stroke("skyblue");   line(x - 7.071, y - 7.071, x + 7.071, y + 7.071);
  } else {                             // |−⟩ – đường chéo \
    stroke("skyblue");   line(x - 7.071, y + 7.071, x + 7.071, y - 7.071);
  }
}
```

c. Chi tiết chức năng BB84 khi có Eve

```
/*****************************************************************
 * MÔ PHỎNG BB84 CÓ KẺ NGHE LÉN (EVE) + TÍNH QBER (p5.js)
 * ---------------------------------------------------------------
 * • Alice gửi n qubit (mặc định 16) cho Bob qua kênh lượng tử.
 * • Eve chèn giữa đường truyền: đo qubit trong cơ sở ngẫu nhiên rồi
 *   gửi lại kết quả đo (đã collapse) cho Bob.
 * • Bob đo qubit nhận được trong cơ sở ngẫu nhiên của mình.
 * • Alice & Bob công khai cơ sở ➜ sàng lọc (sifting) ➜ tạo khóa lượng tử.
```

```
 * • Tính QBER (Quantum Bit Error Rate) để phát hiện sự
hiện diện của Eve.
 * • Tương tác:
 *      – Chuột trái : chạy lại mô phỏng  (init()).
 *      – Chuột phải : chuyển chế độ hiển thị  (ký hiệu
Dirac ↔ biểu tượng).

**********************************************************
**********/


/* ================= BIẾN TOÀN CỤC ================= */
let n        = 16;              // Số qubit mỗi lần mô
phỏng
let dirac    = false;           // Chế độ hiển thị:
                                //   false = biểu tượng
|   true = ký hiệu Dirac
let alice    = [[], [], []];    // [bit gốc, cơ sở mã
hóa, qubit]
let eve      = [[], [], []];    // [cơ sở đo, qubit nhận,
bit Eve đo]
let bob      = [[], [], []];    // [bit đo,  cơ sở đo,
qubit nhận]
let dd       = [];              // dd[i] = true  nếu
Alice & Bob dùng cùng cơ sở
let keyAlice = "";              // Khóa sau sàng lọc
(Alice)
let keyBob   = "";              // Khóa sau sàng lọc
(Bob)
let keyEve   = "";              // Khóa Eve suy đoán (chỉ
để thống kê)
let qber     = 0;               // Quantum Bit Error Rate


/* ============ p5.js – KHỞI TẠO CANVAS ============ */
function setup() {
```

```
  createCanvas(650, 450);       // Kích thước cửa sổ đồ
họa
  init();                       // Sinh dữ liệu ban đầu
}

/* =========== SINH DỮ LIỆU QUANTUM MỚI =========== */
function init() {
  /* Cài đặt thông số vẽ */
  textAlign(CENTER, CENTER);
  strokeWeight(2);

  /* Đặt lại mọi biến toàn cục */
  n        = 16;
  dirac    = false;
  alice    = [[], [], []];
  eve      = [[], [], []];
  bob      = [[], [], []];
  dd       = [];
  keyAlice = "";
  keyBob   = "";
  keyEve   = "";
  qber     = 0;

  /* Biến đếm để tính QBER */
  let siftedCount = 0;   // Tổng số bit giữ lại sau sàng
lọc
  let errorCount  = 0;   // Số bit khác nhau giữa Alice &
Bob

  /* -------- VÒNG LẶP XỬ LÝ n QUBIT -------- */
```

```
  for (let i = 0; i < n; i++) {


    /* 1 ALICE TẠO BIT & CƠ SỞ MÃ HÓA */

    alice[0].push(random([0, 1]));                 // Bit
gốc ngẫu nhiên

    alice[1].push(random([                         // Cơ sở
mã hóa:

      [[1, 0], [0, 1]],                            //   Z-
basis  (|0⟩,|1⟩)

      [["√2/2", "√2/2"], ["√2/2", "-√2/2"]]       //   X-
basis  (|+⟩,|−⟩)

    ]));

    alice[2].push(alice[1][i][alice[0][i]]);     // Trạng
thái qubit tương ứng


    /* 2 EVE CHÈN VÀO ĐƯỜNG TRUYỀN */

    eve[0].push(random([                           // Eve
chọn cơ sở đo

      [[1, 0], [0, 1]],

      [["√2/2", "√2/2"], ["√2/2", "-√2/2"]]

    ]));

    eve[1].push(m(alice[2][i], eve[0][i]));      // Trạng
thái sau đo của Eve

    eve[2].push(cm(eve[1][i], eve[0][i]));       // Bit
Eve suy ra (0/1)

    // Eve gửi qubit đã collapse (theo bit Eve đo) cho
Bob

    let qubitToBob = eve[0][i][eve[2][i]];


    /* 3 BOB NHẬN & ĐO QUBIT */

    bob[1].push(random([                           // Bob
chọn cơ sở đo

      [[1, 0], [0, 1]],
```

```
      [["√2/2", "√2/2"], ["√2/2", "-√2/2"]]
    ]));
    bob[2].push(m(qubitToBob, bob[1][i]));      // Trạng
thái sau đo
    bob[0].push(cm(bob[2][i], bob[1][i]));      // Bit
Bob thu được


    /* 4 SÀNG LỌC (SIFTING): GIỮ BIT KHI CƠ SỞ TRÙNG */
    let sameBasis = (alice[1][i][0][0] ===
bob[1][i][0][0]);
    dd.push(sameBasis);                         // Lưu
lại kết quả


    if (sameBasis) {                            // Nếu
cùng cơ sở:
      keyAlice += alice[0][i];                  //   •
Thêm bit vào khóa Alice
      keyBob   += bob[0][i];                    //   •
Thêm bit vào khóa Bob
      keyEve   += eve[2][i];                    //   •
(Thống kê) khóa Eve
      siftedCount++;                            //   •
Tăng bộ đếm bit hợp lệ
      if (alice[0][i] !== bob[0][i]) errorCount++; //   •
Đếm lỗi nếu có
    }
  }


  /* 5 TÍNH QBER = (Số bit lỗi) / (Số bit giữ lại) */
  qber = (siftedCount > 0) ? errorCount / siftedCount :
0;
}
```

```
/* ============== VÒNG VẼ LIÊN TỤC ================ */
function draw() {
  background(0);            // Làm mới nền (đen)
  textSize(16);
  fill(255);                // Chữ trắng

  /* ----- HIỂN THỊ THEO 2 CHẾ ĐỘ ----- */
  if (dirac) renderDirac();         // Ký hiệu chuỗi / ma trận
  else       renderSymbolic();      // Biểu tượng hình học

  /* ----- HIỂN THỊ KHÓA & QBER (khi rê chuột sang phải) ----- */
  if (mouseX > 525) {
    push();
    strokeWeight(0);
    fill(255);
    text("Alice's Key: " + keyAlice, 175, 425);
    text("Bob's Key: "   + keyBob,   475, 425);
    text("QBER: " + nf(qber, 0, 4),  325, 425);
    pop();
  }
}

/* ========== HÀM HIỂN THỊ CHẾ ĐỘ DIRAC ========== */
function renderDirac() {
  stroke(0);
  for (let i = 0; i < n; i++) {
    let y = (i + 0.75) * (375 / n);        // Hàng i
```

```
    if (mouseX < 625 || dd[i]) {           // Chỉ hiện
khi cần
      text("()=>", 125, y);                // Biểu tượng
qubit bay
      textSize(8);
      text("[" + alice[1][i] + "]", 225, y); // Cơ sở
Alice
      text("[" + eve[1][i]   + "]", 275, y); // Cơ sở Eve
      text("[" + bob[1][i]   + "]", 325, y); // Cơ sở Bob
      textSize(16);
      text("[<>]", 425, y);                // Kênh truyền
      text("Key", 525, y);                 // Cột khóa

      /* ------- TƯƠNG TÁC THEO VỊ TRÍ CHUỘT ------- */
      fill("red");
      if      (mouseX < 125)  text("?", mouseX, y);
// Chưa xác định
      else if (mouseX < 225)  text(alice[0][i], mouseX,
y);         // Bit Alice
      else if (mouseX < 275){ textSize(12);
// Qubit Alice
                              text("[" + alice[2][i] +
"]", mouseX, y);
                              textSize(16); }
      else if (mouseX < 325){ textSize(12);
// Kết quả Eve
                              text("[" + eve[2][i]   +
"]", mouseX, y);
                              textSize(16); }
      else if (mouseX < 425){ textSize(12);
// Qubit Bob
```

```
                                text("[" + bob[2][i]   +
"]", mouseX, y);
                                textSize(16); }
      else {                     text(bob[0][i], mouseX,
y);          // Bit Bob
                                text(alice[0][i], 75, y); }
// Bit Alice
      fill(255);
    }
  }
}



/* ======== HÀM HIỂN THỊ BIỂU TƯỢNG HÌNH HỌC ======== */
function renderSymbolic() {
  for (let i = 0; i < n; i++) {
    let y = (i + 0.75) * (375 / n);

    if (mouseX < 525 || dd[i]) {
      noFill();
      stroke("pink");
      triangle(135, y, 115, y + 10, 115, y - 10);     //
Mũi tên qubit

      /* ------ CỘT CƠ SỞ CỦA ALICE ------ */
      drawBasis(225, y, alice[1][i]);

      /* ------ CỘT CƠ SỞ CỦA EVE ------ */
      drawBasis(275, y, eve[0][i]);

      /* ------ CỘT CƠ SỞ CỦA BOB ------ */
```

```
      drawBasis(325, y, bob[1][i]);

      /* ------ BIỂU TƯỢNG KÊNH VÀ KHÓA ------ */
      stroke("pink");
      quad(435, y, 425, y + 10, 415, y, 425, y - 10);
      stroke("yellow");
      ellipse(525, y, 15, 15);
      line(533, y, 550, y);                  // Răng khóa
      line(545, y - 5, 545, y);
      line(550, y - 5, 550, y);

      /* ------ HIỂN THỊ THEO VỊ TRÍ CHUỘT ------ */
      push();
      if      (mouseX < 125) { /* không hiển thị */ }
      else if (mouseX < 225)  showBit(mouseX, y,
alice[0][i]);          // Bit Alice
      else if (mouseX < 275)  drawState(mouseX, y,
alice[2][i]);     // Qubit Alice
      else if (mouseX < 325)  drawState(mouseX, y,
eve[1][i]);          // Qubit Eve
      else if (mouseX < 425)  drawState(mouseX, y,
bob[2][i]);       // Qubit Bob
      else showBit(mouseX, y, bob[0][i], alice[0][i]);
// Bit Bob + Alice

      /* Chấm tròn đi theo chuột để dễ nhận biết */
      if (mouseX < 625) ellipse(mouseX, y, 20);
      pop();
    }
  }
}
```

```
/* --------- VẼ CƠ SỞ (Z/X) TẠI HOÀNH ĐỘ X --------- */
function drawBasis(x, y, basis) {
  if (basis[0][0] === 1) {               // Z-basis (đỏ)
    stroke("red");
    line(x, y - 10, x, y + 10);          // |
    line(x - 10, y, x + 10, y);          // –
  } else {                               // X-basis
(xanh)
    stroke("skyblue");
    line(x - 7.071, y - 7.071, x + 7.071, y + 7.071); //
/
    line(x - 7.071, y + 7.071, x + 7.071, y - 7.071); //
\
  }
  rect(x - 10, y - 10, 20, 20);          // Khung vuông
}

/* ------ Hiển thị bit (và bit Alice ở cột đầu) ------ */
function showBit(x, y, bitBob, bitAlice = null) {
  push();
  strokeWeight(0);
  fill(255);
  text(bitBob, x, y);          // Bit (Bob hoặc Alice)
  if (bitAlice !== null) text(bitAlice, 75, y); // Bit
gốc Alice (cột đầu)
  pop();
}

/* ------ HÀM VẼ TRẠNG THÁI QUBIT BẰNG BIỂU TƯỢNG ------
*/
```

```
function drawState(x, y, state) {
  if (state[0] === 1) {                        // |1⟩ (đỏ đứng)
    stroke("red");     line(x, y - 10, x, y + 10);
  } else if (state[0] === 0) {                 // |0⟩ (đỏ ngang)
    stroke("red");     line(x - 10, y, x + 10, y);
  } else if (state[1] === "√2/2") {            // |+⟩ (xanh chéo /)
    stroke("skyblue"); line(x - 7.071, y - 7.071, x + 7.071, y + 7.071);
  } else {                                     // |−⟩ (xanh chéo \)
    stroke("skyblue"); line(x - 7.071, y + 7.071, x + 7.071, y - 7.071);
  }
}


/* ======================================================
 *  HÀM m(state, basis) –  Mô phỏng phép đo lượng tử
 *  ----------------------------------------------------
 *  • Nếu state ∈ basis  → trả lại chính state.
 *  • Nếu state ∉ basis  → collapse ngẫu nhiên sang
 *    một trong hai vector của basis.
 * ====================================================== */
function m(state, basis) {
  return (JSON.stringify(basis).includes(JSON.stringify(state)))
         ? state
         : random(basis);
}
```

```
/* ====================================================
 *  HÀM cm(state, basis) –  Chuyển trạng thái thành bit
 *  ----------------------------------------------------
 *  • Nếu khớp vector thứ 1 trong basis ➜ bit 0
 *  • Nếu khớp vector thứ 2             ➜ bit 1
 *  • Nếu không khớp (đo sai cơ sở)     ➜ bit ngẫu nhiên
 * ====================================================
*/
function cm(state, basis) {
  let pos =
JSON.stringify(basis).indexOf(JSON.stringify(state));
  if      (pos > 5) return 1;        // vector thứ 2
  else if (pos > 0) return 0;        // vector thứ 1
  else              return random([0, 1]);
}


/* ========== SỰ KIỆN CHUỘT ========== */
function mousePressed() {
  if (mouseButton === LEFT)  init();      // Chạy lại mô
phỏng
  else                       dirac = !dirac; // Đổi chế
độ hiển thị
}
```

B. Mô phỏng giải thuật BB84

a. Giải thích file qkd_bb84_base

```
"""
-----------------------------------------------------------
-----------------
```

```
MÔ PHỎNG GIAO THỨC BB84 (phiên bản cơ bản – không nhiễu, không tấn công)

------------------------------------------------------------------------

• Định nghĩa các lớp & phương thức để mô phỏng BB84 bằng lập trình hướng đối tượng (OOP) với NumPy.

• Có thể import file này vào một script Python khác trong cùng thư mục.

------------------------------------------------------------------------

Tác giả mà tôi tham khảo : DHRUV BHATNAGAR

------------------------------------------------------------------------
"""


# ____________________________________________________________________

# 1. Thư viện cần thiết

# ____________________________________________________________________

import numpy as np          # Tính toán ma trận/véc-tơ

import random               # Dùng cho vài phép ngẫu nhiên đơn giản


# ____________________________________________________________________

# 2. LỚP alice_con  – Mô hình hoá người gửi (Alice)

# ____________________________________________________________________

class alice_con:
```

```
    """
    Thuộc tính chính sau khi khởi tạo
    ---------------------------------
    s_length      : độ dài chuỗi qubit sẽ gửi
    state_matrix  : ma trận 2×s_length, mỗi cột là trạng thái |ψ⟩ của 1 qubit
    key_arr       : mảng lưu khoá sàng lọc cuối cùng của Alice
    """


    def __init__(self, s_length: int):
        self.s_length     = s_length
        self.state_matrix = np.zeros((2, self.s_length))
# Khởi tạo ma trận trạng thái
        self.key_arr      = np.empty(1)
# Sẽ cập nhật sau


    # 2.1. Sinh ngẫu nhiên dãy cơ sở & dãy bit của Alice
    def seq_init_alice(self):
        self.basis_seq_a  = np.random.randint(0, 2,
self.s_length)  # 0 = Z-basis, 1 = X-basis
        self.states_seq_a = np.random.randint(0, 2,
self.s_length)  # 0/1 tương ứng |0⟩/|1⟩


    # 2.2. Tạo luồng qubit dựa trên 2 dãy trên
    def generate_qubit_stream(self):
        ket0 = np.array([1.0, 0.0])          # |0⟩
        ket1 = np.array([0.0, 1.0])          # |1⟩
        k    = 1.0 / np.sqrt(2.0)
        H    = np.array([[k, k], [k, -k]])   # Cổng
Hadamard H
```

```
        for i in range(self.s_length):
            basis = self.basis_seq_a[i]      # 0 hoặc 1
            state = self.states_seq_a[i]     # 0 hoặc 1

            # Xác định trạng thái qubit:
            if   basis == 0 and state == 0:  # Z-basis & bit 0 → |0⟩
                self.state_matrix[:, i] = ket0
            elif basis == 0 and state == 1:  # Z-basis & bit 1 → |1⟩
                self.state_matrix[:, i] = ket1
            elif basis == 1 and state == 0:  # X-basis & bit 0 → H|0⟩ = |+⟩
                self.state_matrix[:, i] = self.hadamard_alice(ket0)
            else:                            # X-basis & bit 1 → H|1⟩ = |−⟩
                self.state_matrix[:, i] = self.hadamard_alice(ket1)

    # 2.3. Hàm Hadamard riêng của Alice (nếu muốn chỉnh sửa sau này)
    def hadamard_alice(self, state_vec):
        k = 1.0 / np.sqrt(2.0)
        H = np.array([[k, k], [k, -k]])
        return np.matmul(H, state_vec)

    # 2.4. Tạo khoá của Alice sau khi biết dãy cơ sở của Bob
    def key_gen_alice(self, bob_basis_seq):
        self.temp_key_arr = np.zeros(self.s_length)
        self.size_of_key  = 0
```

```
        for i in range(self.s_length):
            if self.basis_seq_a[i] == bob_basis_seq[i]:
# Chỉ giữ bit khi trùng cơ sở
                self.temp_key_arr[i] =
self.states_seq_a[i]
                self.size_of_key    += 1
        # Cắt bỏ phần thừa (chỉ giữ lại size_of_key phần
tử đầu)
        self.key_arr = self.temp_key_arr[:
self.size_of_key].astype(int)


    # 2.5. In khoá (đủ) của Alice
    def print_key_alice(self):
        print("Alice's full key: ", end="")
        for bit in self.key_arr:
            print(int(bit), end="")
        print(f"\nSize of Alice's (full) key is:
{self.size_of_key}")


    # 2.6. So sánh một tập bit kiểm tra với Bob để ước
lượng lỗi
    def key_check(self, check_bits_bob):
        self.size_of_check_bits = len(check_bits_bob)
        self.check_bits_alice   = self.key_arr[:
self.size_of_check_bits]
        self.rem_key_alice      =
self.key_arr[self.size_of_check_bits :]
        self.errors             =
np.sum(self.check_bits_alice != check_bits_bob)


        # Thống kê
        self.percent_error_rate = 100 * self.errors /
self.size_of_check_bits
```

```
        self.key_efficiency     = 100 * self.size_of_key
/ self.s_length

        print("Differences at positions:",
np.bitwise_xor(self.check_bits_alice, check_bits_bob))

        return self.key_efficiency,
self.percent_error_rate


    # 2.7. In phần khoá còn lại (sau khi bỏ bit kiểm tra)

    def print_rem_key_alice(self):

        print("Alice's remaining key: ", end="")

        for bit in self.rem_key_alice:

            print(int(bit), end="")

        print(f"\nSize of Alice's remaining key is:
{len(self.rem_key_alice)}")


    # 2.8. In thông tin debug của Alice

    def print_alice_info(self):

        print("Stream length             :",
self.s_length)

        print("Random basis sequence (A) :",
self.basis_seq_a)

        print("Random state sequence (A) :",
self.states_seq_a)

        print()


    # 2.9. Trả ma trận trạng thái để kênh lượng tử truyền
đi

    def return_state(self):

        return self.state_matrix
```

```
#
_____________________________________________________________
_____________

# 3. LỚP bob_con  – Mô hình hoá người nhận (Bob)

#
_____________________________________________________________
_____________

class bob_con:
    """
    Thuộc tính chính
    ----------------
    basis_seq_b      : dãy cơ sở đo của Bob (0 / 1)
    meas_seq_b       : kết quả đo (bit) của Bob với MẠNG
PHÙ HỢP
    size_of_key      : số bit giữ lại sau sàng lọc
    key_arr          : mảng khoá (Bob)
    """


    def __init__(self, s_length: int):
        self.s_length   = s_length
        self.size_of_key = 0
        self.key_arr     = np.empty(1)


    # 3.1. Nhận ma trận trạng thái từ kênh lượng tử
    def obtain_state(self, state_matrix):
        self.state_matrix = state_matrix


    # 3.2. Sinh ngẫu nhiên dãy cơ sở đo của Bob
    def seq_init_bob(self):
        self.basis_seq_b    = np.random.randint(0, 2,
self.s_length)
```

```
        self.meas_seq_b     = np.zeros(self.s_length)  # Kết quả đo cuối
        self.temp_meas_seq_b = np.zeros(self.s_length) # Lưu tạm từng bước


    # 3.3. Đo toàn bộ luồng qubit
    def meas_qubit_stream_bob(self):
        for j in range(self.s_length):
            st = self.state_matrix[:, j]
            if self.basis_seq_b[j] == 0:                # Đo trong Z-basis
                self.temp_meas_seq_b[j] = self.meas_single_qubit_bob(st)
            else:                                       # Đo trong X-basis
                st = self.hadamard_bob(st)              # Chuyển sang cơ sở Z bằng H
                self.temp_meas_seq_b[j] = self.meas_single_qubit_bob(st)
        self.meas_seq_b = self.temp_meas_seq_b.astype(int)


    # 3.3.1. Đo một qubit (trả về bit 0/1)
    def meas_single_qubit_bob(self, state_vec):
        a     = np.abs(state_vec[0]) ** 2    # Xác suất đo được |0⟩
        thres = 1e-3                         # Ngưỡng so sánh số thực
        # Xử lý 4 trường hợp rìa & ngẫu nhiên theo phân bố Bernoulli(a)
        if np.abs(a - 0) < thres:
            return 1
        elif np.abs(a - 1) < thres:
```

```
            return 0
        elif np.random.uniform() < a:
            return 0
        else:
            return 1

    # 3.3.2. Cổng Hadamard của Bob
    def hadamard_bob(self, state_vec):
        k = 1.0 / np.sqrt(2.0)
        H = np.array([[k, k], [k, -k]])
        return np.matmul(H, state_vec)

    # 3.4. Tạo khoá Bob sau khi biết dãy cơ sở của Alice
    def key_gen_bob(self, alice_basis_seq):
        self.temp_key_arr = np.zeros(self.s_length)
        for i in range(self.s_length):
            if self.basis_seq_b[i] == alice_basis_seq[i]:
                self.temp_key_arr[i] = self.meas_seq_b[i]
                self.size_of_key    += 1
        self.key_arr = self.temp_key_arr[:
self.size_of_key].astype(int)

    # 3.5. In khoá (đủ) của Bob
    def print_key_bob(self):
        print("Bob's full key: ", end="")
        for bit in self.key_arr:
            print(int(bit), end="")
        print(f"\nSize of Bob's (full) key is:
{self.size_of_key}")
```

```
    # 3.6. Chọn ngẫu nhiên một nửa khoá để so sánh công
khai (check bits)
    def pre_key_check(self):
        self.size_of_check_bits =
int(np.ceil(self.size_of_key / 2))
        self.check_bits         = self.key_arr[:
self.size_of_check_bits]
        self.rem_key_bob        =
self.key_arr[self.size_of_check_bits :]
        return self.check_bits


    # 3.7. In phần khoá còn lại của Bob
    def print_rem_key_bob(self):
        print("Bob's remaining key: ", end="")
        for bit in self.rem_key_bob:
            print(int(bit), end="")
        print(f"\nSize of Bob's remaining key is:
{len(self.rem_key_bob)}")



#
_____
_____________

# 4. LỚP q_channel  – Kênh lượng tử (truyền qubit)
#
_____
_____________

class q_channel:
    """
    Hiện tại kênh được mô phỏng lý tưởng (không nhiễu).
    Nếu muốn mô phỏng lỗi, có thể sửa corrupt_state().
    """
```

```
    def get_state(self, st_mtx):
        """Nhận ma trận trạng thái từ Alice."""
        self.st_mtx_in = st_mtx

    def corrupt_state(self):
        """Chèn nhiễu / lỗi vào đây (hiện tại:
identity)."""
        self.out_st_mtx = self.st_mtx_in.copy()

    def put_state(self):
        """Trả ma trận trạng thái cho Bob."""
        return self.out_st_mtx



#
_____
_____

# 5. LỚP c_channel  – Kênh công khai cổ điển (trao đổi
classical)
#
_____
_____

class c_channel:
    """
    Kênh classical dùng để:
      • Trao đổi dãy cơ sở
      • So sánh một phần khoá (check bits) kiểm lỗi
    """

    # 5.1. Trao đổi cơ sở
```

```
    def get_basis_seq(self, basis_seq):
        self.basis_seq = basis_seq

    def put_basis_seq(self):
        return self.basis_seq

    # 5.2. Trao đổi check bits
    def get_check_bits(self, check_bits):
        self.check_bits = check_bits

    def put_check_bits(self):
        return self.check_bits

    # 5.3. Reset kênh (dọn dẹp dữ liệu cũ)
    def ch_reset(self):
        self.basis_seq  = []
        self.check_bits = []
```

b. Giải thích file qkd_experiment_base

```
# Nhập các thành phần mô phỏng giao thức BB84 từ module qkd_bb84_base
import qkd_bb84_base as bb
import random  # (Có thể được dùng trong các lớp bên trong như alice_con hay bob_con)


# Định nghĩa lớp mô phỏng một thí nghiệm QKD dựa trên giao thức BB84
class qkd_experiment:


    def __init__(self, SIZE_TX):
```

```
        # Hàm khởi tạo. SIZE_TX là độ dài chuỗi bit ban đầu mà Alice sẽ sử dụng
        self.SIZE_TX = SIZE_TX
    def build_phase(self):
        # Giai đoạn xây dựng hệ thống: tạo các thực thể đại diện cho Alice, Bob và các kênh truyền
        # Tạo đối tượng Alice với chuỗi bit độ dài SIZE_TX
        self.a0 = bb.alice_con(self.SIZE_TX)
        # Tạo đối tượng Bob tương ứng
        self.b0 = bb.bob_con(self.SIZE_TX)


  # Tạo kênh truyền classical (có thể bị nghe lén nhưng không làm thay đổi dữ liệu)
        self.c_c = bb.c_channel()
        # Tạo kênh truyền quantum (cho phép truyền các qubit)
        self.q_c = bb.q_channel()
    def run_phase(self):
        # Giai đoạn truyền và đo qubit
        # Alice khởi tạo chuỗi bit và cơ sở đo ngẫu nhiên
        self.a0.seq_init_alice()
        # Alice tạo ra dòng qubit tương ứng từ chuỗi bit và cơ sở
        self.a0.generate_qubit_stream()
        # Qubit được truyền đến kênh lượng tử
        self.q_c.get_state(self.a0.return_state())
        # (Trong môi trường không nhiễu nên hàm này không làm gì, nhưng ở môi trường nhiễu thì có thể gây lỗi)
        self.q_c.corrupt_state()
        # Bob khởi tạo chuỗi cơ sở đo ngẫu nhiên của mình
        self.b0.seq_init_bob()
```

```
        # Bob nhận các trạng thái qubit từ kênh lượng tử
        self.b0.obtain_state(self.q_c.put_state())
        # Bob đo từng qubit theo cơ sở của mình
        self.b0.meas_qubit_stream_bob()
    def key_generation_phase(self):
        # Giai đoạn tạo khóa sau khi qubit đã được đo
        # Alice và Bob trao đổi cơ sở đo của nhau qua
kênh classical
        self.c_c.get_basis_seq(self.b0.basis_seq_b)  #
Nhận cơ sở đo của Bob
        self.a0.key_gen_alice(self.c_c.put_basis_seq())
# Alice tạo khóa từ các bit có cơ sở trùng
        self.a0.print_key_alice()  # In khóa của Alice
        self.c_c.ch_reset()  # Đặt lại kênh để chuẩn bị
cho lượt trao đổi tiếp theo


        self.c_c.get_basis_seq(self.a0.basis_seq_a)  #
Nhận cơ sở đo của Alice
        self.b0.key_gen_bob(self.c_c.put_basis_seq())  #
Bob tạo khóa của mình
        self.b0.print_key_bob()  # In khóa của Bob
        self.c_c.ch_reset()
    def validation_phase(self):
        # Giai đoạn kiểm tra độ chính xác của khóa
        # Bob chọn một số bit ngẫu nhiên trong khóa để
kiểm tra với Alice
        self.c_c.get_check_bits(self.b0.pre_key_check())
        # Alice so sánh bit kiểm tra và tính tỷ lệ lỗi
        self.key_efficiency, self.calc_perc_error =
self.a0.key_check(self.c_c.put_check_bits())
        # In khóa cuối cùng còn lại sau khi loại bỏ bit
kiểm tra
```

```
        self.a0.print_rem_key_alice()

        self.b0.print_rem_key_bob()

        # In hiệu suất và tỷ lệ lỗi

        print("Key efficiency obtained is ",
self.key_efficiency, "%")

        print("Qubit error-rate calculated by Alice is ",
self.calc_perc_error, "%")

        self.c_c.ch_reset()

    def execute(self):

        # Hàm tổng hợp để thực hiện toàn bộ quá trình từ
đầu đến cuối

        self.build_phase()           # Khởi tạo các đối
tượng và kênh truyền

        self.run_phase()             # Alice gửi qubit
đến Bob và Bob đo

        self.key_generation_phase()  # Trao đổi cơ sở đo
và tạo khóa

        self.validation_phase()      # Kiểm tra khóa và
tính hiệu suất
```

c. Giải thích file qkd_eve

```
# Nhập các module cần thiết

import qkd_bb84_base as bb             # Thư viện chứa
các lớp cơ bản của hệ thống QKD BB84 (Alice, Bob,
Channel...)

import qkd_experiment_base as expt     # File mô phỏng
BB84 gốc không có Eve

import numpy as np

import random


# Lớp kế thừa từ q_channel để mô phỏng hành vi của kênh
lượng tử bị nghe lén bởi Eve

class eve_q_channel(bb.q_channel):
```

```
    def corrupt_state(self):
        # Nhận luồng trạng thái (qubit) đầu vào từ Alice
        self.temp_st_mtx = self.st_mtx_in
        self.length = np.shape(self.temp_st_mtx[0])[0]

        # Eve tạo một chuỗi cơ sở đo ngẫu nhiên (giống như Bob) để chọn cơ sở đo 0/1 hoặc +/- (Hadamard)
        self.basis_seq_e = np.random.randint(2, size=self.length)

        # Eve sẽ đo từng qubit một
        for i in range(self.length):
            st = self.temp_st_mtx[:, i]  # Lấy vector trạng thái của qubit thứ i
            print("In Eve: input state: ", st)

            # Nếu cơ sở đo là 0 -> cơ sở Z (|0⟩, |1⟩), nếu là 1 -> cơ sở Hadamard (|+⟩, |-⟩)
            if self.basis_seq_e[1] == 0:
                print("Measurement basis: 0/1")
            else:
                print("Measurement basis: +/-")

            if self.basis_seq_e[1] == 0:
                # Đo trong cơ sở chuẩn
                res = self.meas_single_qubit_e(st)
                if res == 0:
                    # Nếu đo được 0, trạng thái sau đo bị sụp đổ về |0⟩ = [1, 0]
```

```
                    self.temp_st_mtx[:, i] = np.array([1,
0])
                else:
                    # Nếu đo được 1, sụp đổ về |1⟩ = [0,
1]
                    self.temp_st_mtx[:, i] = np.array([0,
1])
                print("Returned state: ",
self.temp_st_mtx[:, i])
                print("----------------------------------
-----")


            else:
                # Nếu đo trong cơ sở Hadamard, đầu tiên
phải chuyển qubit sang cơ sở Hadamard
                st = self.hadamard_e(st)
                res = self.meas_single_qubit_e(st)
                if res == 0:
                # Sau khi đo được |+⟩, Eve trả về trạng
thái |+⟩ trong cơ sở ban đầu
                    self.temp_st_mtx[:, i] =
self.hadamard_e(np.array([1, 0]))
                else:
                    # Đo được |−⟩, chuyển về lại cơ sở ban
đầu
                    self.temp_st_mtx[:, i] =
self.hadamard_e(np.array([0, 1]))
                print("Returned state: ",
self.temp_st_mtx[:, i])
                print("----------------------------------
-----")
```

```
        # Gán ma trận trạng thái đầu ra đã bị Eve đo và
can thiệp
        self.out_st_mtx = self.temp_st_mtx


    def meas_single_qubit_e(self, state_vec):
        # Hàm đo xác suất cho trạng thái lượng tử (xác
suất xuất hiện bit 0)
        a = np.square(np.absolute(state_vec[0]))  # a là
xác suất đo được |0⟩ hoặc |+⟩
        print("In Eve: probability of |0> or |+>", a)


        thres = 1e-3  # Ngưỡng sai số


        # Đưa ra kết quả đo lượng tử (0 hoặc 1)
        if np.absolute(a - 0) < thres:
            res = 1
        elif np.absolute(a - 1) < thres:
            res = 0
        elif random.random() < a:
            res = 0
        else:
            res = 1
        return res


    def hadamard_e(self, state_vec):
        # Hàm thực hiện phép biến đổi Hadamard lên vector
trạng thái đầu vào
        k = 1.0 / np.sqrt(2.0)
        H = np.array([[k, k], [k, -k]])  # Ma trận
Hadamard
        return np.matmul(H, state_vec)
```

```
# Kế thừa lớp mô phỏng ban đầu để thay kênh lượng tử bằng kênh bị nghe lén
class eve_qkd_experiment(expt.qkd_experiment):


    def build_phase(self):
        # Gọi hàm build ban đầu để tạo Alice, Bob, classical channel...
        super().build_phase()


        # Gán lại kênh lượng tử là phiên bản có Eve
        q_c_e = eve_q_channel()
        self.q_c = q_c_e  # Gán kênh lượng tử đã bị thay đổi vào hệ thống



def main():
    SIZE_TX = 200  # Độ dài của chuỗi bit được gửi


    # Tạo đối tượng mô phỏng BB84 có sự can thiệp của Eve
    e1 = eve_qkd_experiment(SIZE_TX)


    # Thực hiện toàn bộ quy trình BB84 (gồm cả build, run, key gen và validation)
    e1.execute()



if __name__ == '__main__':
    main()
```

d. Mã mô phỏng kịch bản một

```
import numpy as np

import matplotlib.pyplot as plt

from qkd_noise_model import noisy_qkd_experiment  # Import lớp mô phỏng QKD có nhiễu (bao gồm fading Gamma-Gamma)

import random


def simulate_fading_performance(sizes, num_trials=30, alpha=3.0, beta=2.5, P_H_FAIL=0.0, fixed_seed=True):

    # Hàm mô phỏng hiệu suất QKD theo kích thước bit truyền với fading

    qber_mean, qber_std = [], []              # Danh sách lưu trung bình và độ lệch chuẩn của QBER

    eff_mean, eff_std = [], []                # Danh sách lưu trung bình và độ lệch chuẩn của hiệu suất khóa

    retained_key_ratios = []                  # Danh sách lưu tỉ lệ giữ lại khóa trung bình


    for size in sizes:

        qbers, effs, retained_ratios = [], [], []  # Kết quả cho từng giá trị SIZE_TX

        for i in range(num_trials):               # Chạy nhiều lần để lấy kết quả trung bình

            if fixed_seed:

                np.random.seed(i)                 # Đặt seed cố định cho numpy để kết quả tái lập

                random.seed(i)                    # Đặt seed cố định cho random


            exp = noisy_qkd_experiment(size, P_H_FAIL, alpha, beta)  # Khởi tạo mô phỏng với tham số fading
```

```
            exp.execute()
# Thực hiện toàn bộ giao thức BB84


            qbers.append(exp.calc_perc_error)
# Tính QBER sau khi lọc bit
            effs.append(exp.key_efficiency)
# Tính hiệu suất khóa (bit còn lại / bit ban đầu)
            retained = len(exp.a0.rem_key_alice) / size
# Tỉ lệ giữ lại khóa sau sàng lọc
            retained_ratios.append(retained)


        qber_mean.append(np.mean(qbers))         # Trung
bình QBER theo size
        qber_std.append(np.std(qbers))           # Độ
lệch chuẩn QBER
        eff_mean.append(np.mean(effs))           # Trung
bình hiệu suất
        eff_std.append(np.std(effs))             # Độ
lệch chuẩn hiệu suất

retained_key_ratios.append(np.mean(retained_ratios))  #
Trung bình tỉ lệ giữ lại khóa


    return {
        "sizes": sizes,
        "qber_mean": qber_mean,
        "qber_std": qber_std,
        "eff_mean": eff_mean,
        "eff_std": eff_std,
        "retained_key_ratio": retained_key_ratios
    }  # Trả về kết quả mô phỏng dưới dạng dictionary
```

```
def find_optimal_size(results, qber_thresh=11.0, min_eff=30.0):
    # Hàm tìm kích thước SIZE_TX tối ưu dựa trên ngưỡng QBER và hiệu suất tối thiểu
    sizes = results["sizes"]
    qbers = results["qber_mean"]
    effs = results["eff_mean"]
    retained = results["retained_key_ratio"]

    candidates = []
    for i in range(len(sizes)):
        if qbers[i] < qber_thresh and effs[i] > min_eff:  # Điều kiện thỏa mãn
            score = effs[i] * retained[i] / (1 + qbers[i])  # Tính điểm theo công thức
            candidates.append((sizes[i], score))  # Ghi lại ứng viên tiềm năng

    if not candidates:
        print("⚠ Không có giá trị SIZE_TX nào thỏa mãn điều kiện!")
        return None

    optimal_size = max(candidates, key=lambda x: x[1])[0]  # Chọn SIZE_TX có điểm cao nhất
    return optimal_size

def plot_results(results):
    # Hàm vẽ biểu đồ QBER, hiệu suất, và tỉ lệ giữ lại theo SIZE_TX
    sizes = results["sizes"]
```

```
    plt.figure(figsize=(15, 5))  # Kích thước tổng thể


    plt.subplot(1, 3, 1)
    plt.errorbar(sizes, results["qber_mean"],
yerr=results["qber_std"], fmt='-o', color='red')
    plt.title("QBER theo số bit truyền")
    plt.xlabel("SIZE_TX")
    plt.ylabel("QBER (%)")
    plt.grid(True)


    plt.subplot(1, 3, 2)
    plt.errorbar(sizes, results["eff_mean"],
yerr=results["eff_std"], fmt='-o', color='green')
    plt.title("Hiệu suất khóa (%)")
    plt.xlabel("SIZE_TX")
    plt.ylabel("Key Efficiency")
    plt.grid(True)


    plt.subplot(1, 3, 3)
    plt.plot(sizes, results["retained_key_ratio"],
marker='o', color='blue')
    plt.title("Tỉ lệ giữ lại khóa")
    plt.xlabel("SIZE_TX")
    plt.ylabel("Tỉ lệ giữ lại (%)")
    plt.grid(True)


    plt.tight_layout()
    plt.show()


if __name__ == "__main__":
```

```
    sizes = [100, 200, 400, 600, 800, 1000, 1200, 1500,
2000]  # Các kích thước bit truyền cần kiểm tra

    results = simulate_fading_performance(sizes,
num_trials=30, alpha=3.0, beta=2.5, P_H_FAIL=0.0,
fixed_seed=True)

    plot_results(results)  # Vẽ biểu đồ kết quả


    optimal = find_optimal_size(results)  # Tìm kích
thước tối ưu

    if optimal:

        print(f"\n✅ SIZE_TX tối ưu được đề xuất là:
{optimal} bits")
```

e. Mã mô phỏng kịch bản hai

```
import numpy as np                          # Thư viện hỗ trợ
tính toán ma trận, trung bình,...

import matplotlib.pyplot as plt             # Thư viện vẽ biểu
đồ

from qkd_noise_model import noisy_qkd_experiment  # Lớp
mô phỏng QKD có nhiễu fading Gamma-Gamma

import random                               # Dùng để tạo ngẫu
nhiên với seed


def simulate_qber_vs_fading(SIZE_TX=400, alphas=[1, 2, 3,
4, 5], betas=[1, 2, 3, 4, 5], num_trials=30,
P_H_FAIL=0.0, fixed_seed=True):

    # Hàm mô phỏng QBER theo các giá trị fading (Gamma-
Gamma): alpha và beta

    qber_matrix = np.zeros((len(alphas), len(betas)))  #
Ma trận lưu trung bình QBER tương ứng (alpha, beta)


    for i, alpha in enumerate(alphas):          # Lặp qua
các giá trị alpha (chỉ số hình dạng của phân bố Gamma-
Gamma)
```

```
        for j, beta in enumerate(betas):        # Lặp qua
các giá trị beta (cũng là chỉ số hình dạng)
            qbers = []                          # Danh
sách lưu QBER của từng lần thử
            for k in range(num_trials):         # Chạy mô
phỏng nhiều lần để lấy kết quả trung bình
                if fixed_seed:
                    np.random.seed(k)           # Đặt
seed cố định để kết quả tái lập
                    random.seed(k)


                exp = noisy_qkd_experiment(SIZE_TX,
P_H_FAIL, alpha, beta)  # Tạo mô phỏng với các tham số
                exp.execute()                   # Thực
hiện toàn bộ giao thức BB84 với fading Gamma-Gamma
                qbers.append(exp.calc_perc_error)  # Lưu
QBER (số lỗi / tổng số bit giữ lại)


            qber_matrix[i, j] = np.mean(qbers)  # Tính
QBER trung bình với (alpha, beta) và lưu vào ma trận


    return qber_matrix, alphas, betas  # Trả về ma trận
QBER và danh sách alpha, beta


def plot_qber_heatmap(qber_matrix, alphas, betas):
    # Hàm vẽ biểu đồ heatmap cho QBER theo alpha và beta
    plt.figure(figsize=(8, 6))  # Kích thước biểu đồ


    plt.imshow(
        qber_matrix,              # Ma trận QBER làm dữ
liệu ảnh
        cmap='viridis',           # Bảng màu
```

```
        origin='lower',           # Gốc tọa độ ở dưới
(thay vì trên)
        aspect='auto',            # Cho phép tự điều
chỉnh tỉ lệ trục
        extent=[min(betas), max(betas), min(alphas),
max(alphas)]  # Định nghĩa trục tọa độ
    )


    plt.colorbar(label='QBER (%)')      # Thêm thanh màu
bên phải biểu diễn giá trị QBER
    plt.xlabel('Beta')                  # Gán nhãn trục X
    plt.ylabel('Alpha')                 # Gán nhãn trục Y
    plt.title('QBER theo Alpha và Beta (Gamma-Gamma
Fading)')  # Tiêu đề biểu đồ
    plt.xticks(betas)                   # Gán nhãn trục X
theo beta
    plt.yticks(alphas)                  # Gán nhãn trục Y
theo alpha
    plt.grid(False)                     # Tắt grid (lưới)
    plt.tight_layout()                  # Canh chỉnh tự
động để không bị tràn mép
    plt.show()                          # Hiển thị biểu
đồ


if __name__ == "__main__":
    SIZE_TX = 400                      # Kích thước khối
bit truyền (bit lượng tử truyền đi)
    alphas = [1, 2, 3, 4, 5]           # Danh sách các
alpha đại diện cho độ nhiễu nhỏ đến lớn
    betas = [1, 2, 3, 4, 5]            # Danh sách các
beta đại diện cho nhiễu Gamma-Gamma
    qber_matrix, alphas, betas = simulate_qber_vs_fading(
        SIZE_TX, alphas, betas, num_trials=30,
fixed_seed=True
```

```
    )                                       # Thực hiện mô phỏng lấy ma trận QBER
    plot_qber_heatmap(qber_matrix, alphas, betas)  # Vẽ biểu đồ heatmap từ kết quả
```

g. Mã mô phỏng kịch bản ba

```
import numpy as np  # Thư viện xử lý mảng và số ngẫu nhiên
from scipy.special import kv as Kv, gamma  # Hàm đặc biệt (chưa sử dụng trong đoạn mã này)


# Hàm tính xác suất lỗi QBER theo cường độ I (do fading Gamma-Gamma gây ra)
def calc_qber(I, threshold=0.2):
    return np.clip(0.5 * np.exp(-I / threshold), 0, 0.5)  # QBER giảm theo I, được giới hạn trong khoảng [0, 0.5]


# Hàm sinh mẫu fading Gamma-Gamma bằng cách nhân 2 biến gamma độc lập
def gamma_gamma_sample(alpha, beta, size):
    X = np.random.gamma(alpha, 1.0, size)  # Mẫu gamma với tham số alpha
    Y = np.random.gamma(beta, 1.0, size)   # Mẫu gamma với tham số beta
    return X * Y  # Nhân lại để có fading Gamma-Gamma


# Mô phỏng BB84 không có sự xuất hiện của kẻ nghe lén (Eve)
def bb84_no_eve(num_bits=10000, alpha=1, beta=1, threshold=0.2):
    alice_bits = np.random.randint(0, 2, num_bits)  # Bit ngẫu nhiên của Alice (0 hoặc 1)
```

```
    alice_basis = np.random.randint(0, 2, num_bits)  # Basis ngẫu nhiên của Alice (0 hoặc 1)
    bob_basis = np.random.randint(0, 2, num_bits)    # Basis ngẫu nhiên của Bob (0 hoặc 1)


    I = gamma_gamma_sample(alpha, beta, num_bits)  # Cường độ fading Gamma-Gamma
    error_probs = calc_qber(I, threshold)  # Tính xác suất lỗi cho từng bit


    bob_bits = np.zeros(num_bits, dtype=int)  # Khởi tạo bit Bob nhận được


    for i in range(num_bits):
        if bob_basis[i] == alice_basis[i]:  # Nếu basis trùng, Bob đo chính xác hơn
            bit = alice_bits[i]
            if np.random.rand() < error_probs[i]:  # Lỗi do fading
                bit = 1 - bit  # Lật bit
            bob_bits[i] = bit
        else:
            bob_bits[i] = np.random.randint(0, 2)  # Nếu basis khác, Bob đo ngẫu nhiên


    sifted_indices = np.where(alice_basis == bob_basis)[0]  # Lọc ra các bit trùng basis
    errors = np.sum(bob_bits[sifted_indices] != alice_bits[sifted_indices])  # Đếm số lỗi
    qber = errors / len(sifted_indices)  # Tính QBER
    return qber, len(sifted_indices)  # Trả về QBER và số bit đã sàng lọc
```

```
# Mô phỏng BB84 có Eve tấn công kiểu intercept-resend
def bb84_with_eve(num_bits=10000, alpha=3, beta=2, threshold=0.2):
    alice_bits = np.random.randint(0, 2, num_bits)  # Bit của Alice
    alice_basis = np.random.randint(0, 2, num_bits)  # Basis của Alice
    eve_basis = np.random.randint(0, 2, num_bits)  # Eve chọn basis ngẫu nhiên
    eve_bits = np.zeros(num_bits, dtype=int)  # Bit Eve nhận

    for i in range(num_bits):  # Eve đo qubit
        if eve_basis[i] == alice_basis[i]:
            eve_bits[i] = alice_bits[i]  # Nếu đo đúng basis thì đo đúng
        else:
            eve_bits[i] = np.random.randint(0, 2)  # Nếu sai basis thì đo ngẫu nhiên

    I = gamma_gamma_sample(alpha, beta, num_bits)  # Fading từ Eve đến Bob
    error_probs = calc_qber(I, threshold)  # Xác suất lỗi khi Eve gửi lại cho Bob

    bob_basis = np.random.randint(0, 2, num_bits)  # Bob chọn basis ngẫu nhiên
    bob_bits = np.zeros(num_bits, dtype=int)  # Bit Bob nhận

    for i in range(num_bits):  # Bob đo dựa trên qubit mà Eve gửi lại
        if bob_basis[i] == eve_basis[i]:
```

```
            bit = eve_bits[i]
            if np.random.rand() < error_probs[i]:
                bit = 1 - bit  # Lỗi do fading
            bob_bits[i] = bit
        else:
            bob_bits[i] = np.random.randint(0, 2)  # Nếu basis sai, đo ngẫu nhiên

    sifted_indices = np.where(alice_basis == bob_basis)[0]  # Lọc bit trùng basis giữa Alice và Bob
    errors = np.sum(bob_bits[sifted_indices] != alice_bits[sifted_indices])  # Đếm lỗi
    qber = errors / len(sifted_indices)  # QBER
    return qber, len(sifted_indices)

# Chạy mô phỏng với các giá trị số bit khác nhau để so sánh ảnh hưởng
bit_values = [1000, 5000, 10000, 20000, 50000]  # Các mức số bit thử nghiệm
qber_no_eve_list = []  # Lưu QBER không có Eve
qber_eve_list = []     # Lưu QBER có Eve

for n in bit_values:
    qber_no_eve, sift_no_eve = bb84_no_eve(num_bits=n)  # Chạy mô phỏng không có Eve
    qber_eve, sift_eve = bb84_with_eve(num_bits=n)      # Chạy mô phỏng có Eve
    qber_no_eve_list.append(qber_no_eve)
    qber_eve_list.append(qber_eve)
    print(f"Số bit: {n}")
    print(f"  QBER không có Eve: {qber_no_eve:.4f} (số bit sàng lọc: {sift_no_eve})")
```

```
    print(f"  QBER có Eve: {qber_eve:.4f} (số bit sàng
lọc: {sift_eve})")
    print("-" * 40)


# Vẽ biểu đồ so sánh QBER giữa có và không có Eve
import matplotlib.pyplot as plt


plt.plot(bit_values, qber_no_eve_list, 'o-', label="QBER
không có Eve")
plt.plot(bit_values, qber_eve_list, 's-', label="QBER có
Eve")
plt.xlabel("Số bit đầu vào")
plt.ylabel("QBER")
plt.title("So sánh QBER BB84 dưới fading Gamma-Gamma")
plt.legend()
plt.grid(True)
plt.show()
```