QryptoTalk: An Interactive Exploration of Quantum Key Distribution

Abstract

We didn’t just build a simulation — we built a conversation. **QryptoTalk** is our attempt to make quantum cryptography not just understandable, but feelable. Through an interactive, step-by-step simulation of the BB84 protocol, we break down how secure communication works at the quantum level — from photons and random bases to shared secret keys and the invisible threat of eavesdroppers.

This wasn’t about proving how much we knew. It was about simplifying what the world barely talks about — quantum safety, in a future where classical encryption could fall apart.

In **QryptoTalk**, users see every moment: Alice’s random bit generation, the fragile photon transmission, Bob’s unpredictable measurements, and the quiet panic when Eve tries to sneak in. We show how mismatches turn into security signals, and how matching bases create trust — bit by bit.

But it doesn’t stop at theory. Our final step walks users through how the shared quantum key is actually used — applying XOR logic for real-world encryption, and highlighting how this isn’t just science fiction anymore.

Whether you’re a beginner or someone chasing quantum research, **QryptoTalk** speaks your language — with colors, clarity, and calm storytelling.

Because we believe knowledge shouldn’t just be read — it should be experienced.

Welcome to a project where photons meet feelings — and where learning finally feels like discovery.

Problem Statement

“Why we built this, and what was missing in the world before it.”

In a world slowly approaching the edge of classical encryption's reliability, quantum cryptography isn’t just a buzzword — it’s the future defense system. But here’s the truth no one wants to say out loud:

Quantum cryptography is **cold, distant, and intimidating.**

It’s locked behind jargon, research papers, and animations that feel more like textbooks than explanations.

We looked around and saw a gap — not in theory, but in emotion.

There was no space where someone could feel what was happening when photons were exchanged, when bases matched, or when Eve quietly tried to break into a conversation that was supposed to be private.

We didn’t want to just explain how **BB84 works**.
We wanted people to **see it. Touch it. Understand it** like they would a story.

So we created **QryptoTalk** — a digital, visual, and emotional walkthrough of the BB84 protocol. Not as a research tool, but as a human-centric learning space.

A place where beginners don’t feel left out, and where even experts pause and say,

"Oh. That’s actually beautiful."

Core Concepts Briefly Explained

“Not just definitions — this is where the science begins to speak.”

What is Quantum Cryptography?

It’s the art of using quantum mechanics to secure information — not through math problems that take a billion years to solve, but through **physics** itself.
Here, even *trying to listen* changes the message. Even *trying to clone* breaks the code.

It’s not about locking the door — it’s about building a system where *the moment someone even touches the handle*, you already know.

What is the BB84 Protocol?

BB84 is the heartbeat of quantum key distribution.
It’s how **Alice** sends quantum bits (qubits) to **Bob**, using **random quantum bases**. Bob tries to measure them with his own random choices, and afterward, they only keep the bits where their choices matched.

Those matching bits become a **shared secret key** — built from nothing but randomness, light, and trust.

What is the Quantum No-Cloning Theorem?

Here’s the rule that makes quantum cryptography bulletproof:

You can’t perfectly copy a quantum state. Not even if you’re a god of computing.

That means if an attacker (like **Eve**) tries to intercept the qubit, she must measure it. And that very act of measuring? It disturbs the qubit. **Alters it. Breaks its trust.**

It’s like trying to copy a whisper — you always leave noise behind.

Why Can’t Eve Copy Qubits?

Because quantum systems don’t behave like files you Ctrl+C and Ctrl+V.
She’d have to guess Alice’s basis. If she guesses wrong, she alters the qubit.
If Bob then measures it later with the correct basis — the error becomes obvious.

Eve can only **observe or preserve**, but **not both**. That’s the paradox she can’t escape.

⚠ How Basis Mismatches Lead to Errors

When Alice and Bob choose **different bases**, Bob’s result becomes **random**. It’s not a bug — it’s quantum truth.

And when Eve enters the picture, she forces even more mismatch.

> And in that chaos... Alice and Bob find their clue.
> They **sacrifice a few bits** to test for discrepancies.
> If the error rate is too high — **they know**. She was there.

Real-World Usage

This isn't just theory locked in labs. Quantum key distribution is being used in:

- **Banking systems** for ultra-secure transactions
- **Government communications** that must stay classified for decades
- **Defense networks** where even a single intercepted bit could mean war
- **Satellites and fiber optic networks** in countries building post-quantum infrastructure

> "This is not some future dream. It's already happening — quietly, globally.
> And with QryptoTalk, we're handing that future to students, learners, and creators — in the most human way we could imagine."

Project Features

"What QryptoTalk does — not just as a tool, but as an experience."

We didn't want to build just another explainer page with floating buzzwords and broken equations.
We wanted you to **see the protocol breathe**, step by step — like a silent dance between trust and uncertainty.
Here's what makes **QryptoTalk** more than a project:

1. The Full BB84 Protocol — Visualized

We broke down the BB84 protocol into **six clear, interactive steps**, from Alice generating random bits all the way to Bob decrypting the final message.

Each section is *not just told*, but **shown**, with color-coded states, movement, and natural flow — making even beginners feel like they understand quantum mechanics intuitively.

2. Eve's Interception Toggle

One button. One decision.
You click, and suddenly **Eve appears** — not as code, but as a force.

You get to *see* what happens when she tries to eavesdrop.
The photons react. Errors appear. The system feels disturbed.

> It's a moment — not just a feature — where the user becomes part of the security check.

3. Real-Time XOR Encryption with the Shared Key

We don't stop at key generation like most simulations.
We go further — all the way to **actual encryption**.

You see how Alice uses the key to encrypt a message using **XOR logic**.
And how Bob decrypts it perfectly with the same key.
Bit by bit. Visually. Honestly. Beautifully.

4. Aesthetic Animations and UI

From glowing photons to smooth floating particles, QryptoTalk is a **space that feels alive**.
Not like a science paper — but like a short animated film made for thinkers.

We cared about how it moves, how it flows, how it feels.
Because the story deserves more than dry code.

5. Educational Flow that Doesn’t Feel Like a Lecture

Each section teaches something. But not in a top-down way. It’s more like a conversation between the protocol and the person using it.

We added:

- Real-world metaphors
- Visual grids for outcomes
- Simple language with professional clarity
- FAQs to answer what the heart wonders but the mind hesitates to ask

Every scroll, every section, every animation — it’s designed to build **clarity and curiosity at the same time**.

If learning ever felt like art — this is what it would look like.

Technical Stack

“How we built the soul beneath the visuals.”

QryptoTalk may feel like magic, but under the hood, it’s carefully crafted with code, logic, and passion.
Here’s what powered our build:

Frontend Development

- **HTML5 & CSS3**: For semantic structure and modern, responsive design
- **Custom CSS Variables & Animations**: Smooth transitions, float, pulse, and photon-move effects
- **JavaScript (Vanilla JS)**: Handling logic, event toggles, FAQ animations, scroll behavior, and UI interactivity
- **Responsive Design**: Mobile-friendly layout, optimized across screens

Interactive Behavior

- **DOM manipulation** to dynamically control Eve's appearance, photon motion, and message encryption
- **Step-by-step simulations** crafted manually — no frameworks — so everything was **learned and written from scratch**

Tools Used

- **VS Code**: Main development environment
- **GitHub**: Version control and project hosting
- **Figma** *(Optional)*: For planning visual layout and information flow
- **InShot**: (used in social material/video, if any)

Optional Future Stack (not used yet, but considered)

- **Qiskit** for backend quantum simulation
- **Streamlit or Flask** for live user interaction
- **Firebase** for saving sessions or real key usage tracking

We chose minimalism for control.
No heavy libraries. No clutter. Just pure, expressive code doing one thing:
Making quantum logic finally feel human.

7. How It Works (Step-by-Step)

“The heartbeat of QryptoTalk — broken down like a story.”

If you slow it down, the BB84 protocol feels less like a technical procedure and more like a **conversation between two minds trying to build trust** — in a world where even the air might be listening.

Here’s how it unfolds, under the hood:

Step 1: Alice Generates Random Bits and Bases

She doesn’t know what Bob will do. She doesn’t need to.
Alice begins by choosing two things randomly:

- A sequence of **bits** (0s and 1s)
- A matching sequence of **bases** (+ or ×)

These are her **private thoughts**, encoded in the language of quantum.

Step 2: Alice Encodes Qubits Using Her Bases

She takes each bit and turns it into a **quantum state**, based on the base she chose:

- In **+** basis:
  0 → |0⟩, 1 → |1⟩
- In **×** basis:
  0 → |+⟩, 1 → |-⟩

Each photon she sends is now a **tiny secret wrapped in light**.

Step 3: Transmission Through the Quantum Channel

She sends the qubits to Bob through a fragile path — usually an **optical fiber** or **free-space**.
Here's the catch:

> These photons are so sensitive that **even looking at them the wrong way changes them forever.**

They reach Bob — but not untouched.

Step 4: Bob Measures Using Random Bases

Bob, without knowing Alice's choices, picks his own bases — randomly.

- If he chooses the **same base** as Alice:
  ✅ He gets the correct bit
- If he chooses the **wrong base**:
  ⚠ He gets a **random bit**, with a 50/50 chance

It’s like trying to tune into someone’s frequency without knowing their channel — and sometimes, it aligns.

Step 5: They Publicly Compare Bases

After transmission, Alice and Bob **reveal only their bases**, not the actual bits.

Wherever their bases match, they **keep the bits**.
Where they differ, the bits are **discarded** — lost to quantum uncertainty.

These matching bits form their **shared raw key**.

Step 6: Key is Used for XOR Encryption

Now that they hold the **same private key**, they encrypt messages using XOR:

- Bit by bit, one line of trust at a time
- Even if someone sees the encrypted message, they can't decrypt it — not without the exact key

> It's not just encryption — it’s **shared silence**, turned into code.

Step 7: Eve Tries to Interfere — and Quantum Mechanics Fights Back

Eve, the invisible listener, intercepts and measures the qubits.

But she can’t clone. She can’t guess the bases.
So, whenever she’s wrong, she introduces **errors** — like fingerprints on glass.

When Alice and Bob compare a sample of their key bits — they notice something's wrong.

And that’s how they know:

"Someone was here who shouldn’t have been."

Eavesdropping & Security

“This is where quantum cryptography fights back.
Not with guns or firewalls — but with the laws of nature itself.”

Why Does Eve Introduce Errors?

Because **she has to guess the basis** for each qubit.
And when she guesses wrong, the quantum state collapses unpredictably.
Even if she tries to act silently — her interference **leaves a trace**.

In quantum systems:

You can’t observe without changing.
You can’t copy without corruption.
You can’t cheat without being caught.

How is Her Presence Detected?

Alice and Bob do something bold:
They **sacrifice a small portion of their key** — and compare it publicly.

- If the error rate is **within limits** (low), they continue.
- If the error rate is **too high**, it’s proof of interception.

Quantum doesn't lie. It exposes the intruder without ever needing to see her face.

What Happens If Too Many Errors Are Found?

Simple: They **discard the entire key**.
They start again — maybe with a new set of bits, maybe with a better channel.

But they **never proceed** with a compromised key.

Because in quantum cryptography, **trust is sacred**.
Even a hint of intrusion is enough to throw the entire plan away.

> And that's what makes this system unbreakable:
> It doesn't just encrypt. It protects **honesty** in the act of communication.

Real-World Relevance

"Why this project doesn't just matter in code — it matters in history."

The world we live in is encrypted — our messages, our bank data, our political systems, even the most private texts we send. But almost all of that is guarded by **classical encryption**, like RSA.

And here's the ticking clock:

> **Quantum computers will break RSA.**
> Not *maybe*. Not *someday*. It's a question of **when**, not **if**.

RSA, ECC, and other classical methods rely on problems like factoring large primes — things that take traditional computers centuries.
But with Shor's Algorithm on a powerful enough quantum machine?

All that breaks. Instantly.

Enter: Quantum Key Distribution (QKD)

QKD doesn't rely on complexity — it relies on **physics**.
It's not about making things *hard to crack*, it's about making them **impossible to touch without detection**.

That's why BB84 and other QKD protocols are already being used by:

- **Chinese quantum satellites**
- **Swiss banking systems**
- **Defense communications in Europe and the US**

This is the **next layer of digital survival**.

Why QryptoTalk Matters

Most people don't know what quantum security even means.
Even fewer understand it visually or emotionally.

QryptoTalk gives people — students, educators, developers — a way to:

- **See how security can be rooted in nature**
- **Understand how eavesdropping detection works**
- **Play with the process** rather than read abstract math

> "It's not a project that just teaches the future.
> It **feels like the future."**

Screenshots + Diagrams

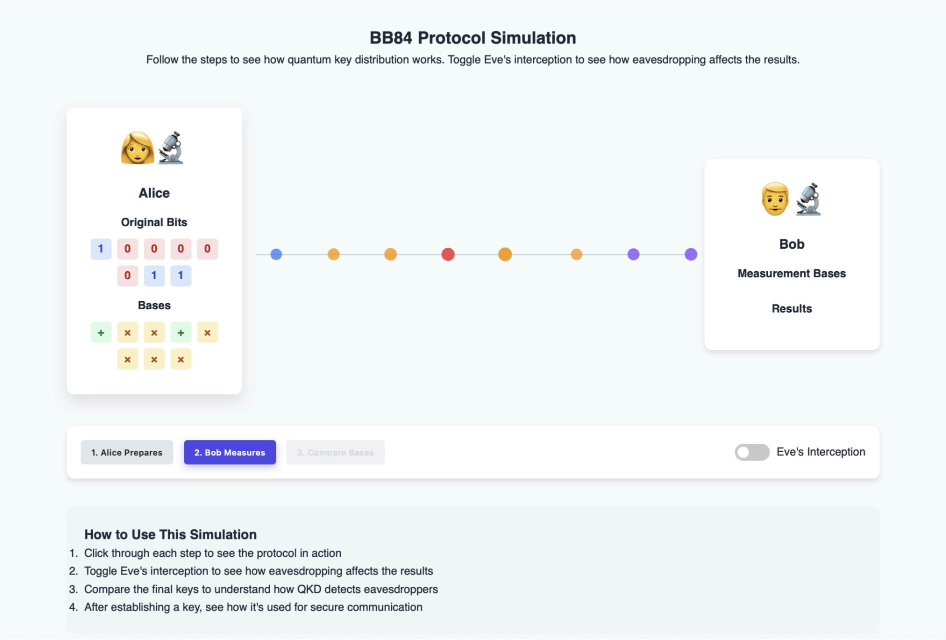
Because sometimes, seeing is the real understanding.

For our documentation and submission, we include the following visuals:

Screenshots of Each BB84 Step

Each simulation step is captured with visual clarity:

1. **Bit + Basis Generation**



BB84 Protocol Simulation

Follow the steps to see how quantum key distribution works. Toggle Eve's interception to see how eavesdropping affects the results.

Alice

Original Bits

1 0 0 0 0
0 1 1

Bases

+ × × + ×
× × ×

Bob

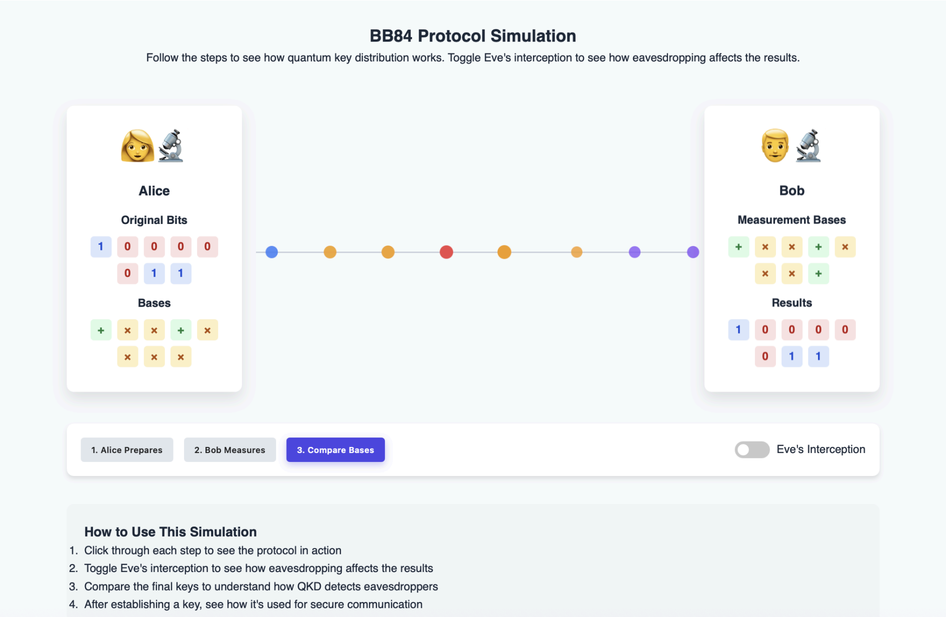
Measurement Bases

Results

How to Use This Simulation

1. Click through each step to see the protocol in action
2. Toggle Eve's interception to see how eavesdropping affects the results
3. Compare the final keys to understand how QKD detects eavesdroppers
4. After establishing a key, see how it's used for secure communication

2. Bob’s Random Measurement



BB84 Protocol Simulation

Follow the steps to see how quantum key distribution works. Toggle Eve's interception to see how eavesdropping affects the results.

Alice

Original Bits

1 0 0 0 0
0 1 1

Bases

+ × × + ×
× × ×

Bob

Measurement Bases

+ × × + ×
× × +

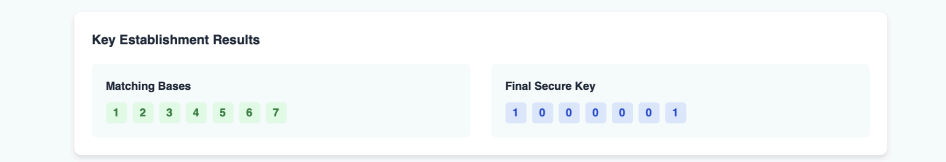
Results

1 0 0 0 0
0 1 1

How to Use This Simulation

1. Click through each step to see the protocol in action
2. Toggle Eve's interception to see how eavesdropping affects the results
3. Compare the final keys to understand how QKD detects eavesdroppers
4. After establishing a key, see how it's used for secure communication

3. Basis Matching & Bit Filtering



Key Establishment Results

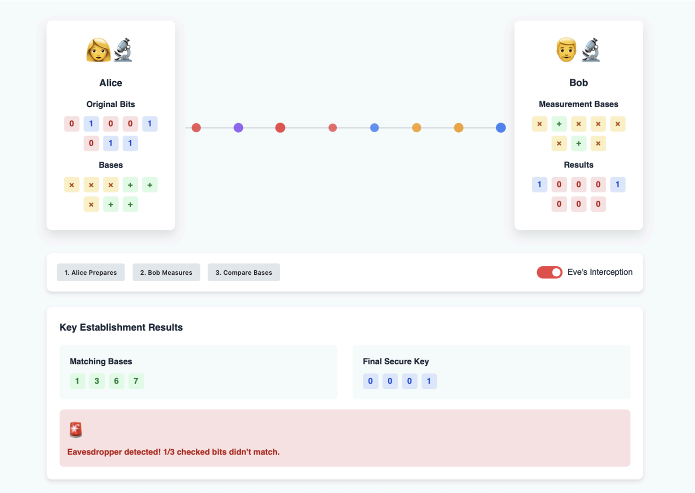
Matching Bases

1 2 3 4 5 6 7

Final Secure Key

1 0 0 0 0 0 1

4. Eavesdropper Detection



Alice

Original Bits

0 1 0 0 1
0 1 1

Bases

× × × + +
× + +

Bob

Measurement Bases

× + × × ×
× + ×

Results

1 0 0 0 1
0 0 0

Key Establishment Results

Matching Bases

1 3 6 7

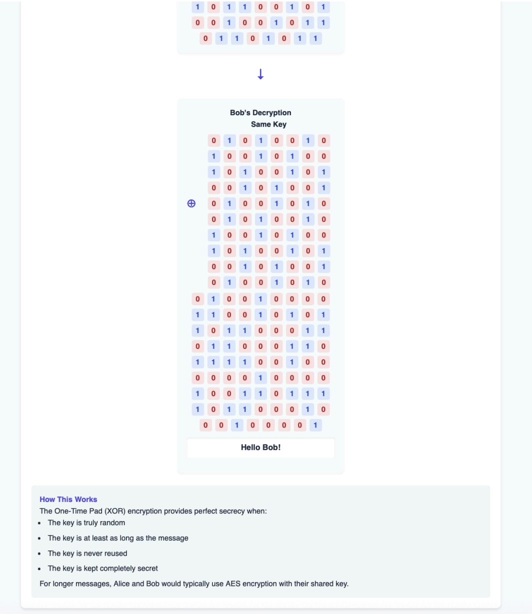
Final Secure Key

0 0 0 1

🚨

Eavesdropper detected! 1/3 checked bits didn't match.

5. **Final Message Encryption/Decryption with Shared Key**



```
1 0 1 1 0 0 1 0 1
0 0 1 0 0 1 0 1 1
 0 1 1 0 1 0 1 1
```

↓

Bob's Decryption
Same Key

```
    0 1 0 1 0 0 1 0
    1 0 0 1 0 1 0 0
    1 0 1 0 0 1 0 1
    0 0 1 0 1 0 0 1
⊕   0 1 0 0 1 0 1 0
    0 1 0 1 0 0 1 0
    1 0 0 1 0 1 0 0
    1 0 1 0 0 1 0 1
    0 0 1 0 1 0 0 1
    0 1 0 0 1 0 1 0
  0 1 0 0 1 0 0 0 0
  1 1 0 0 1 0 1 0 1
  1 0 1 1 0 0 0 1 1
  0 1 1 0 0 0 1 1 0
  1 1 1 1 0 0 1 0 0
  0 0 0 0 1 0 0 0 0
  1 0 0 1 1 0 1 1 1
  1 0 1 1 0 0 0 1 0
   0 0 1 0 0 0 0 1
```

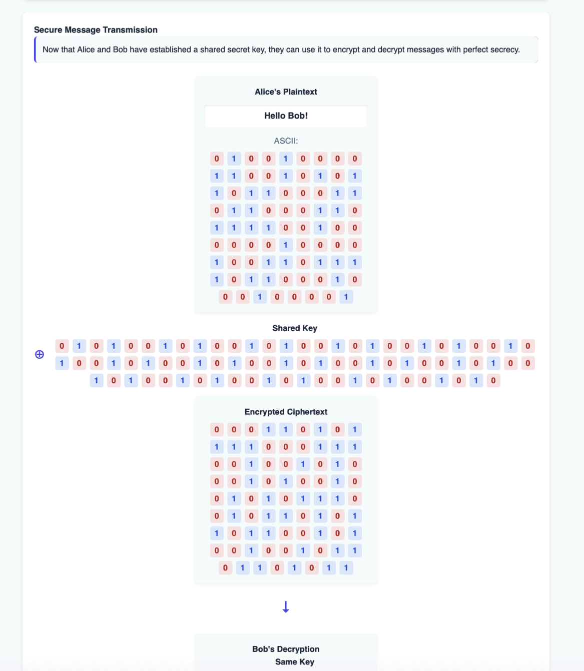
Hello Bob!

How This Works

The One-Time Pad (XOR) encryption provides perfect secrecy when:

- The key is truly random
- The key is at least as long as the message
- The key is never reused
- The key is kept completely secret

For longer messages, Alice and Bob would typically use AES encryption with their shared key.



Secure Message Transmission

> Now that Alice and Bob have established a shared secret key, they can use it to encrypt and decrypt messages with perfect secrecy.

Alice's Plaintext

Hello Bob!

ASCII:

```
0 1 0 0 1 0 0 0 0
1 1 0 0 1 0 1 0 1
1 0 1 1 0 0 0 1 1
0 1 1 0 0 0 1 1 0
1 1 1 1 0 0 1 0 0
0 0 0 0 1 0 0 0 0
1 0 0 1 1 0 1 1 1
1 0 1 1 0 0 0 1 0
 0 0 1 0 0 0 0 1
```

Shared Key

```
  0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0
⊕ 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0
      1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0
```

Encrypted Ciphertext

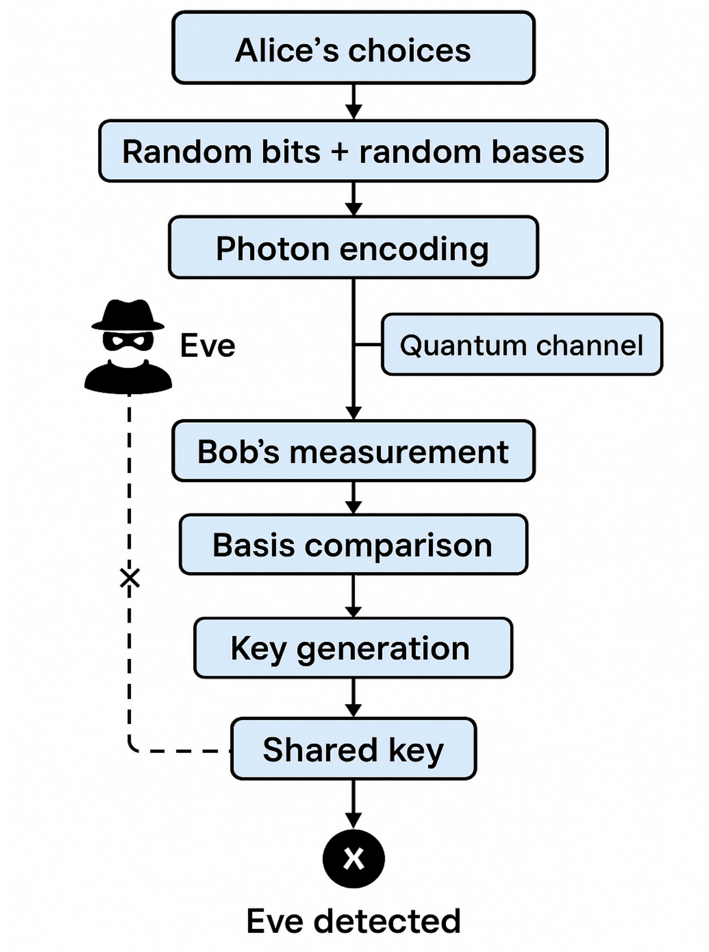
```
0 0 0 1 1 0 1 0 1
1 1 1 0 0 0 1 1 1
0 0 1 0 0 1 0 1 0
0 0 1 0 1 0 0 1 0
0 1 0 1 0 1 1 1 0
0 1 0 1 1 0 1 0 1
1 0 1 1 0 0 1 0 1
0 0 1 0 0 1 0 1 1
 0 1 1 0 1 0 1 1
```

↓

Bob's Decryption
Same Key

Each image is captioned and color-coded to make even the most technical step intuitive.

BB84 Protocol Flowchart



A complete diagram showing:

- Alice’s choices
- Photon encoding
- Bob’s measurement
- Basis comparison
- Key generation
- Eve detection logic

Clean, printable, and suitable for teaching.

Classical vs Quantum Key Exchange Comparison Chart

Feature	Classical (RSA)	Quantum (BB84)
Depends on Math?	✅	❌
Broken by Quantum Computers?	❌	✅
Eavesdropper Detection?	❌	✅
Based on Physics?	❌	✅
Future-Proof?	❌	✅

This gives the reader a **one-glance understanding** of why QKD is revolutionary.

How to Use It

> **“For judges, users, or curious learners — this is how to explore QryptoTalk.”**

How to Run the Simulation (Local or Hosted)

- You can open the site directly via our **hosted link**:
  🔗 `qryptotalk.vercel.app`

- Or clone the GitHub repo

Interactive Controls and Buttons

Feature	What it does
“Try Simulation” Button	Takes user to the full BB84 protocol walkthrough
“Toggle Eve” Button	Simulates an eavesdropper interfering with qubit transmission
Hover/Click on Steps	Reveals matching bases, discarded bits, and sample keys
XOR Encryption Preview	Shows encrypted and decrypted messages using the shared key

Step-by-Step Guide (What Each Part Teaches)

1. **Hero Section** – Introduction to quantum security

2. **Process Timeline** – Detailed BB84 protocol walkthrough

3. **Visual Interactions** – Eve appears, error triggers, key bits compared

4. **Message Encryption** – Real XOR example with animation

5. **FAQ Section** – Answers to common doubts with real-world relevance

This isn't just a website you click through.
It's a space you explore — like a short film wrapped in logic.

Future Scope

> **"What this project could become — if given more time, more energy, and more imagination."**

QryptoTalk was never meant to be the final word.
It was meant to be a beginning — a door you open, not a conclusion you read.

Here's where we know this can go next:

1. Qiskit Backend Integration

Right now, our qubits are visual metaphors. But we could integrate IBM's **Qiskit SDK** to simulate **real quantum circuits**, making the backend *authentically quantum* — not just educational.

2. Support for Other Protocols (BB92, E91, etc.)

While BB84 is foundational, adding:

- **BB92**: Fewer states, different logic

- **E91**: Entanglement-based QKD

...would open the simulation to **next-gen quantum experiments** — and show how protocols evolve with new physics.

3. User-Driven Encryption Input

Let users type in a custom message and watch it **transform into binary**, encrypted via their generated key in real-time.

> This would turn the platform from a simulator into a **personal encryption sandbox.**

4. Photon Noise Simulation

Introduce simulated **channel noise**, showing how real-world conditions like fiber degradation affect QKD.

Users could adjust:

- Photon loss rate
- Basis mismatch probability
- Eve’s level of aggression

And observe **how key reliability drops under stress** — just like in reality.

5. Classroom Mode / Teacher Toolkit

Let educators run this simulation in “presentation mode,” walking students through each part with **guided commentary, pauses, and editable bit streams**.

The future is big. But this foundation is already solid.
It’s just waiting for the next quantum leap.

What I Learned

> **“This wasn’t just code — it was a journey. And every bug, every diagram, every late night taught something bigger than syntax.”**

Technically:

- I learned **how BB84 works at the deepest level**, not just in theory but as a flow of logic I had to visualize, animate, and explain.
- I understood how **quantum randomness**, **no-cloning**, and **error detection** are not just physics quirks — they’re *security guarantees*.
- I built this entire simulation using **HTML, CSS, and pure JavaScript** — no frameworks, no shortcuts.
- I practiced **front-end design**, **state transitions**, **DOM manipulation**, and the art of **explaining complexity without oversimplifying**.

Personally:

- I learned the value of **clarity over complexity** — that teaching something hard is harder than just doing it.
- I understood what it means to work on something that feels **bigger than a grade or a prize**.
- I saw how code could carry not just logic, but *emotion* — how animations could teach, and how photons could tell

stories.

- I struggled, but I didn't stop — and now I know I can build anything I truly care about.

> "**QryptoTalk** is the most honest thing I've ever built.
> Because it reflects what I love: science, simplicity, storytelling."

Credits, Acknowledgments & About the Author

Created by:
Abhinaw Singh
Student · Dreamer · Builder

I'm a curious soul who doesn't just study science — I *feel* it. From quantum mechanics to consciousness, from notebooks to neural networks, my work is where logic meets love, and education meets emotion.

I believe learning should be **experiential**. That **code can tell stories**, and even the most complex ideas can feel **beautiful** when shared with clarity and care.

QryptoTalk is one of my many efforts — not to impress, but to **illuminate**. I'm not here just to build projects.
I'm here to build **understanding** — one photon at a time.

Tools Used:

- **HTML / CSS / JavaScript** – for building the full simulation

- **Visual Studio Code** – my coding canvas

- **GitHub** – version control and deployment
- **Google Fonts (Inter)** – for clean, minimal typography
- **Stack Overflow & MDN Web Docs** – trusted allies during bugs and brain fog
- **My own notes, questions, and late-night voice memos** – the raw foundation of it all