Summer 2021 QKD Research Summary

Leo Crowder

August 2021

1 Netsquid Simulation

My netsquid simulation and comments can be accessed in the jupyter notebook file QKD_wavelengths_decoys.ipynb. The file allows you to set a list of states to use for QKD and a (possibly empty) list of decoy state which Bob can check for, as well as parameters like depolarization loss, and raw key length. Because the QKD network is modeled with netsquid, specific characteristics about the physical components can be added (quantum channel noise models, quantum memory Pauli noise, decoherence, etc.) [1]

1.1 Network Setup

Refer to figures 1 and 2 for the two different network setups, dependent on the truth value of `eve_present`. Note that the protocol runs on a clock in Alice's node, which dictates when qubits are emitted from Alice's quantum source. My simulation has no delay model, and the addition of a delay model may require some tweaks to the protocol.

1.2 "Wavelength Qubit" Subclass

Netsquid does not have a built-in attribute for wavelength, so in cell 3 I defined the `WavelengthQubit` subclass of `ns.qubits.qubit.Qubit`, and a function `make_wavelength_qubit` which takes a qubit and a wavelength, and creates a `WavelengthQubit` object out of this.

1.3 Protocol Steps

1. The clock in Alice's node ticks, sending a signal to trigger Alice's quantum source.
2. A $|0\rangle$ is emitted and stored in Alice's quantum memory.
3. Alice randomly selects a basis and signed bit value from the list of allowed states, and operates on the qubit to transform into the desired state.
4. Once prepared, Alice sends the qubit through the quantum channel.
5. If Eve is present, she intercepts the qubit by waiting until it arrives in her quantum memory; once this has happened, Eve randomly selects a basis, performs a measurement, and sends the qubit along to Bob.
6. Bob awaits an input from the quantum channel into his quantum memory. Once he has received the qubit, he also selects a basis and measures the qubit.

7. Repeat for desired number of ticks `n`.

Note:

- Basis information is labeled as 0s (rectilinear basis) and 1s (diagonal basis)
- "Signed bits" refers to the way netsquid assigns a 0 or 1 to a measurement outcome based on the sign of the eigenvalue associated with the measurement result. So in the rectilinear basis, $0°$ polarization maps to 0 and $90°$ polarization maps to 1. In the diagonal basis, $45°$ ($|+\rangle$) maps to 0 and $135°$ ($|-\rangle$) maps to 1. This is useful because it is an easy way to keep track of the quantum states being sent/observed/received, but is distinguished from the actual logical bit (or trit) value that a particular state is assigned.
- The previous two points make for an easy way to turn a $|0\rangle$ state into one of the desired states for QKD, and is what I use in the simulation: let b denote the basis label, s the signed bit value, H the Hadamard operator, X the Pauli X operator and I the identity matrix. Then the desired quantum state is

$$[bH + (1-b)I][sX + (1-s)I]|0\rangle$$

2 Formulas

2.1 Depolarization Error Model

To model errors in the quantum channel, I used a time-independent depolarization error model, in which you assign the probability p that qubits depolarize, causing the original state ρ (density matrix) to depolarize into the mixed state

$$\varepsilon(\rho) = \frac{1}{2}pI + (1-p)\rho$$

Within the scope of states being prepared in the simulation, this results in an error rate of $p_e = \frac{1}{2}p$. Thus all of the following formulas with error rate can be alternatively expressed with depolarization probability.

2.2 Formula table

Refer to figure 3 for a visual explanation of the ternary decoy-state QKD protocol.

- $QBER =$ qubit error rate
- $d_s =$ proportion of decoy states in sifted key (qubits with matching bases)
- $d_r =$ proportion of decoy states in raw key (all qubits Bob receives)

Protocol	No Eve	Eve
BB84	$QBER = p_e$	$QBER = \frac{1}{2}p_e + \frac{1}{4}$
Ternary decoy-state QKD	$d_s = \frac{1}{3}p_e$ $d_r = \frac{1}{6}p_e + \frac{1}{6}$	$d_s = \frac{1}{6}p_e + \frac{1}{12}$ $d_r = \frac{1}{12}p_e + \frac{5}{24}$

I often drew probability trees [2] to derive these formulas, as it made it easiest to visualize. Simulation results agreed with these formulas, as shown in figure 4.

3 Statistics

3.1 Hypothesis Testing Intro

Hypothesis testing is a way to determine whether or not a particular data set follows an expected distribution, and is how I analyzed the security of different QKD protocols. [3]

In BB84, the data set is the sifted key, and we are concerned with $QBER$. Suppose the expected error rate is p_e. Then in a sifted key of length n, we expect to see np_e errors. If the actual error rate turns out to be $q \neq np_e$, we want to determine if this difference is a result of reasonable statistical deviations or comes from a different distribution (for example, if there is an eavesdropper). The same can be done for the decoy-state protocol but with d_s or d_r instead of $QBER$.

3.2 Decoy-state QKD hypothesis testing setup

Suppose Alice and Bob expect an error rate of p_e due to depolarization in their quantum channel. Then from the formula table in 2.2, Bob expects to see a proportion $d_0 = \frac{1}{3}p_e$ of decoy states in their sifted key if there is no eavesdropping. Suppose their sifted key has length n. By performing the protocol and generating this sifted key, they are sampling from a binomial distribution

$$\begin{aligned} P(k) &= \binom{n}{k} d_0^k (1-d_0)^{n-k} \\ &= \binom{n}{k} \left(\frac{1}{3}p_e\right)^k \left(1 - \frac{1}{3}p_e\right)^{n-k} \end{aligned}$$

where $P(k)$ is the probability Bob observes k decoys in the sifted key and n is the sifted key length.

An eavesdropper will induce errors which increases the proportion of decoy states. In the worst case scenario, if Eve has advanced technology which allows her to prevent/correct depolarization which Alice and Bob expect, then she will induce a decoy rate of $\frac{1}{6}(0) + \frac{1}{12} = \frac{1}{12}$. Thus, as long as Alice and Bob's expected decoy rate is below $\frac{1}{12}$ (equivalently, if $QBER < \frac{1}{4}$; refer to formula table in 2.2), then they can attempt to determine the presence of Eve with hypothesis testing.

Let d_s be the actual proportion of decoy states in Alice and Bob's sifted key when they implement the protocol. Then Alice and Bob's null and alternative hypotheses are

$$H_0: \quad d_s = d_0$$

$$H_a: \quad d_s > d_0$$

i.e., the null hypothesis is the decoy rate expected from no eavesdropping, and the alternative hypothesis is anything greater. From here they would pick a significance level α and perform a hypothesis test. Refer to (SOURCE) for how the test can actually be done, they explain it better than I will be able to.

3.3 Power Analysis Setup

A power analysis allows you to compute the "statistical power" of a hypothesis test [4]. Note the meaning of statistical parameters α and β in context of QKD:

$$\begin{aligned} \alpha &= \text{significance level} \\ &= \text{the probability of rejecting a true null hypothesis} \\ &= \text{the probability that Bob detects eavesdropping when there was in fact no eavesdropper} \end{aligned}$$

$$\begin{aligned}\beta &= 1 - \text{power}\\ &= \text{the probability of not rejecting a false null hypothesis}\\ &= \text{the probability that Bob detects no eavesdropping when there was in fact an eavesdropper}\end{aligned}$$

Thus, power is essentially the security of the protocol (the probability that Bob detects eavesdropping when there is an eavesdropper). Furthermore, to perform a power analysis we specify an alternative hypothesis (this is NOT done in a hypothesis test), which is the scenario in which Eve eavesdrops. This is useful because we can specifically compare the Eve/No Eve distributions, and compute how confident we can be in our ability to "distinguish" the distributions from one another. To increase power we can:

- increase α
- increase the sample size (key length) n

The bulk of my statistical analysis was in the form of power tests, with the goal of comparing how different protocols answer questions such as

- How many bits (sample size) must Bob check for at least 99% confidence (power) in the security of the protocol with error rate p_e?
- Given a key length (sample size) and significance level α, what is the best security level (power) Alice and Bob can expect to have?
- What is the longest secret key length Alice and Bob can achieve given some minimum power, raw key length, an error rate p_e, and significance level α.

*Note: In a decoy-state protocol, Alice and Bob do not need to sacrifice a subset of their sifted key to publicly compare for errors (as in BB84). Bob can privately check the entire sifted key for decoy states, meaning the entire sifted key is the sample size. In BB84, the sample size is simply however many bits Alice and Bob choose to compare, and is therefore less than the sifted key length if Alice and Bob are going to reserve bits for their encryption key.

3.4 Power Analysis Results

I performed power analyses numerically using `decoy_QKD_power_analysis.py` with $\alpha = 0.1$. Figures 5-7 show plots of various results. Some takeaway concepts:

- Suppose $p_e = 0$. Observe the difference in $QBER$ when Eve is not present ($QBER = 0$) versus when she is present ($QBER = 0.25$). Likewise, observe the difference in d_s when Eve is not present ($d_s = 0$) versus when she is present ($d_s \simeq 0.083$). The difference is even smaller in the raw key decoy rate: $d_r \simeq 0.167$ with no Eve versus $d_r \simeq .208$ with Eve present.
- This difference is important because it is harder to distinguish distributions with means that are closer together. Thus, to get the same confidence in BB84 and the decoy-state protocol, we need to check more bits in the decoy-state protocol. (Refer to figure (?????) for an idea of how many more bits are needed)

4 Eavesdropping Strategies

I spent quite some time exploring eavesdropping strategies for the ternary decoy-state protocol, and found that, given some knowledge of the trit mapping algorithm, Eve can learn a third of the states without making any quantum measurements (and therefore inducing no additional errors).

4.1 Explanation

Suppose Eve measures the wavelength of all qubits Alice sends, which she can do without perturbing the polarization. Furthermore, suppose she learns

- Alice's basis choices, which will be public if Alice and Bob wish to sift their key, and
- the states Alice will use as decoy states (the states that Alice does not send).

I will now refer to the two wavelengths by colors, red and green (as in figure 3), since it is easier to explain. If the green decoy state is 90° polarization as in figure 3, then Eve knows that all green qubits in the rectilinear basis must be 0°, since Alice never sends the 90° decoy state. Likewise, if the red decoy state is 135° polarization, then Eve knows all red qubits prepared in the diagonal basis are 45° polarization. Thus, Eve knows 1/3 of the states Alice has sent. If Eve also learns the whole trit mapping algorithm (which states map to +,-,0), she then knows the trit values of 1/3 of the key.

4.2 Mitigation

This problem was the reason that I also investigated the efficacy of Bob checking the entire raw key for decoy states. Because, if Alice never makes her basis choices public, then Eve cannot gain information via the method described above. Although I considered it worthwhile and included it in my investigation, I suspect a major flaw in this idea is that basis reconciliation might be inevitable in order to sift the key and remove errors. Thus I only investigated this method within statistical analyses of security, and not as a means of generating a secret key.

Additionally, and perhaps most importantly, there is apparently a way to mitigate this problem that involves some classical cryptographic/computational techniques which are far outside my realm of knowledge.

References

[1] Netsquid Documentation: https://docs.netsquid.org/latest-release/

[2] Probability Trees: https://www.statisticshowto.com/how-to-use-a-probability-tree-for-probability-questions/

[3] Hypothesis Testing: https://en.wikipedia.org/wiki/Statistical_hypothesis_testing

[4] Power Analysis: https://machinelearningmastery.com/statistical-power-and-power-analysis-in-python/

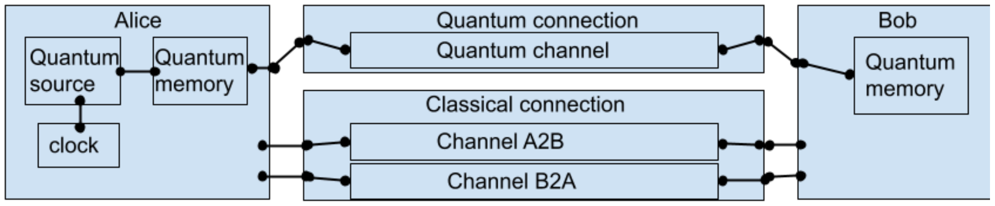


Figure 1: QKD network with no eavesdropper

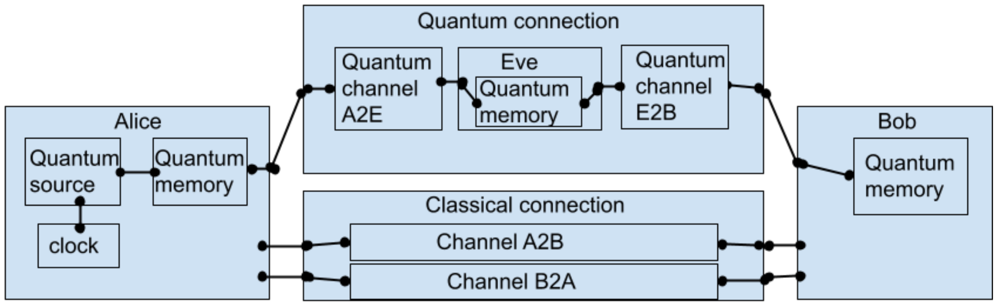
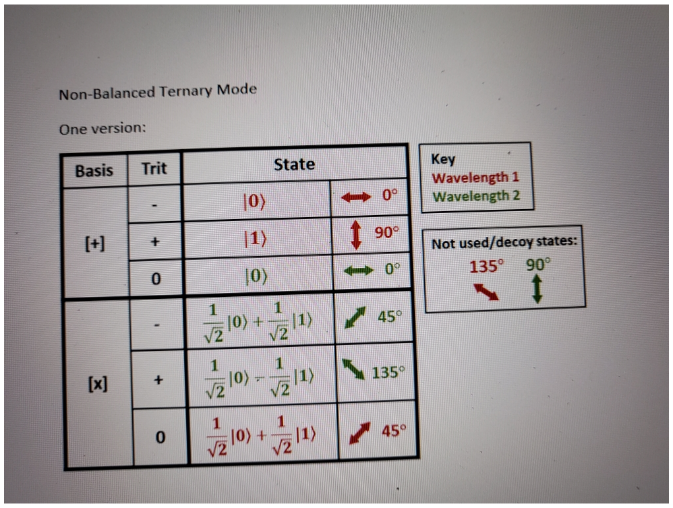


Figure 2: QKD network with Eve eavesdropping



Non-Balanced Ternary Mode

One version:

Basis	Trit	State	
[+]	-	$\vert 0\rangle$	↔ 0°
	+	$\vert 1\rangle$	↕ 90°
	0	$\vert 0\rangle$	↔ 0°
[x]	-	$\frac{1}{\sqrt{2}}\vert 0\rangle + \frac{1}{\sqrt{2}}\vert 1\rangle$	↗ 45°
	+	$\frac{1}{\sqrt{2}}\vert 0\rangle - \frac{1}{\sqrt{2}}\vert 1\rangle$	↖ 135°
	0	$\frac{1}{\sqrt{2}}\vert 0\rangle + \frac{1}{\sqrt{2}}\vert 1\rangle$	↗ 45°

Key
Wavelength 1
Wavelength 2

Not used/decoy states:
135° ↘ 90° ↕

Figure 3: Polarizations and trit assignments of ternary decoy-state QKD

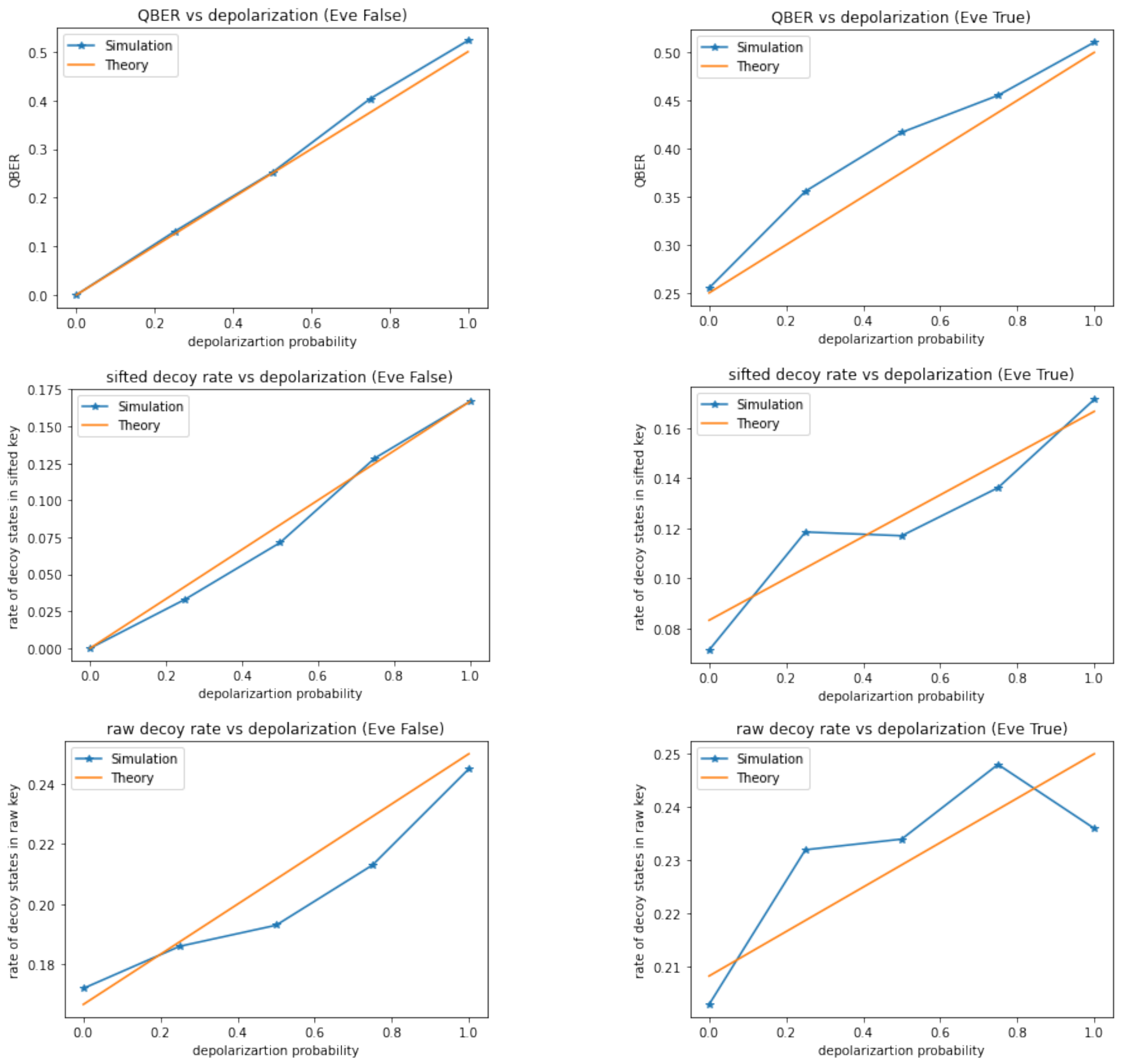


Figure 4: Netsquid simulation results (raw key length=1000) of the non-balanced ternary QKD protocol with decoy states, as shown in figure 3. Note that these are plotted against depolarization probability instead of error rate. 'Theory' lines are taken directly from the table of derived formulas in section 2.2.

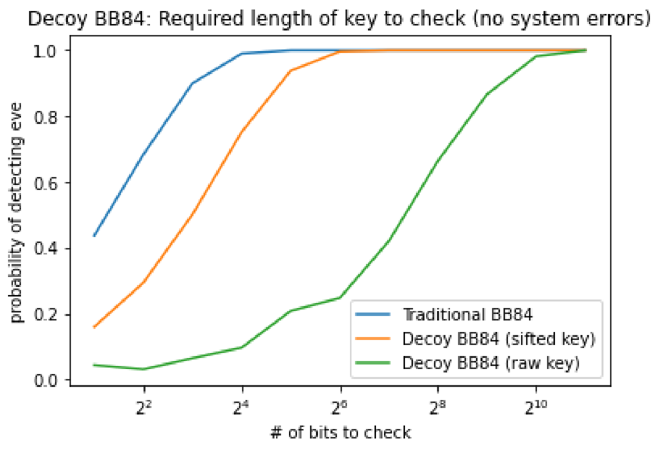


Figure 5: Power analysis results. Different protocols require a larger sample size (number of bits) to achieve a high probability of detecting Eve. (Remember that probability of detecting eve is equal to power.)

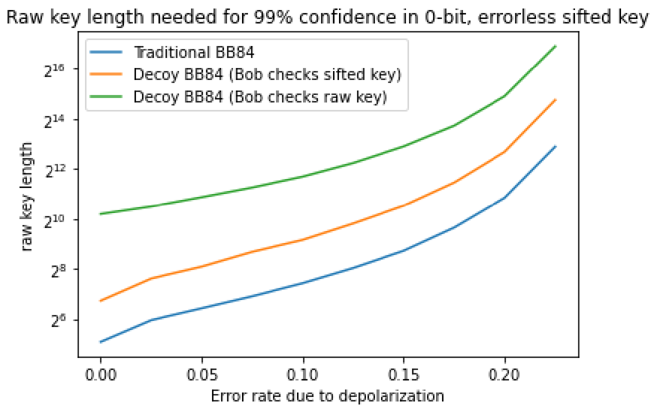


Figure 6: Power analysis results. Raw key length required to reach 99% power for various error rates. This excludes the reservation of bits for the final secret key in BB84 (hence the "0-bit key" in the title) to allow for more direct comparison.

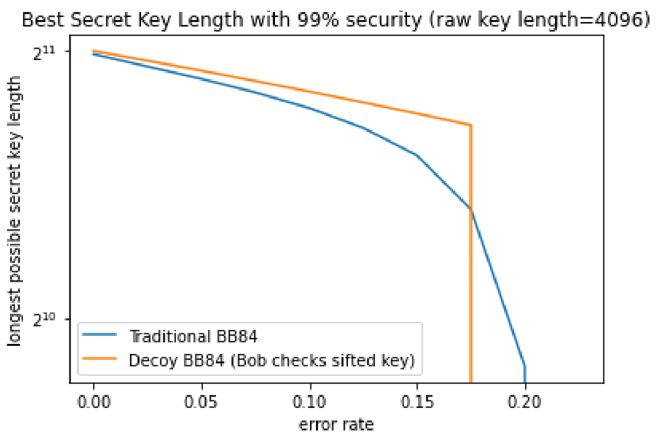


Figure 7: Power analysis results. Given the number of qubits Alice can send (raw key length) is 2^{12}, this plot shows the longest secret key Alice and Bob can generate, while still meeting the 99% power threshold. Observe the vertical lines: this is where Alice and Bob "run out" of bits, meaning they no longer have a large enough sample size to meet the 99% power minimum.

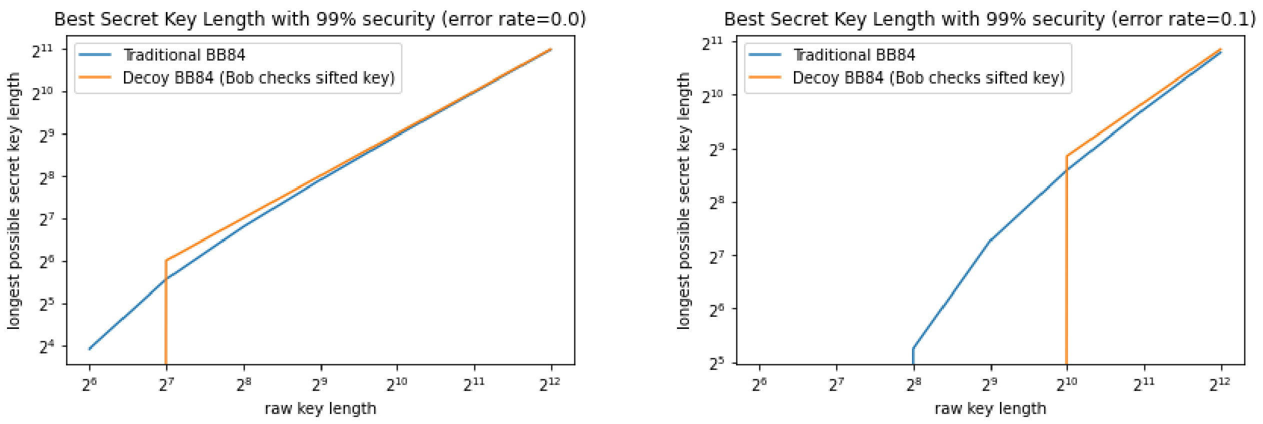


Figure 8: Power analysis results. Now we see the dependence of "longest secret key" on raw key length. Curves are zero where the raw key length is not sufficient for the desired security level. The decoy state protocol requires a minimum key length that is longer than BB84 because it requires a larger sample size. Notice, however, that once this minimum is achieved, the decoy-state protocol can always generate secret keys that are longer than BB84, since no bits are sacrificed for comparison. Observe that with increased error, both thresholds are shifted to the right as both protocols require larger sample sizes.