

Study and optimization of energetic consumption of a post-processing chain for Continuous Variable Quantum Key Distribution

Damien ASSIRE

Supervisors: Adrien CASSAGNE and Yoann PIÉTRI

Contents

1	Introduction	2
1.1	Context	2
1.2	Previous work	2
2	Overview of a quantum key distribution protocol	2
2.1	Privacy amplification	3
2.2	Information reconciliation	5
2.3	Secret key rate	5
2.4	Libraries used	6
3	Experiments	6
3.1	Platform	6
3.2	Methodology	7
4	Results	8
4.1	First experiments	8
4.2	Privacy amplification	8
4.3	Information reconciliation	10
5	Conclusion	12
5.1	Perspective	13
6	Bibliography	14

1 Introduction

1.1 Context

When two parties want to communicate with each other, cryptography allows them to do so without any other one to access the information being communicated. Classical cryptography relies on hard algorithmic problems, and the assumption that an opponent have a limited computational power and won't be able to solve those problems in a certain amount of time. However, with improvement in classical hardware and in quantum computing, those assumptions may not stay with time. Quantum key distribution in contrast, offers information-theoretic security: the amount of information an opponent can get is bounded, even with unlimited computational power and time.

Two main families of QKD protocols exists: the continuous variable protocols and the discrete variable protocols. In CV-QKD protocols, information is encoded in quadrature of the electromagnetic field, which are measured by the receiver with homodyne or heterodyne detection. Those protocols can be used with standard optical telecommunication components, like lasers and coherent state detectors, whereas DV-QKD protocols need other components like single-photon detectors. However, this come at the cost of a more complex post-processing steps.

A question that emerged in the field of quantum technologies is the energy they used [Auf22]. Where this question has seen some work in the field of quantum computing, few has been done for quantum communication, including quantum key distribution. In this work, we studied the energetic aspects of the post-processing of a CV-QKD protocol, which has never been studied.

1.2 Previous work

The energetic aspects of a quantum key distribution has already been a subject. A previous work [Yeh+24] studied the physical layer and the digital signal processing part, which also introduced a metric, the energy efficiency, which corresponds to the amount of bits we can expect for a given amount of energy:

$$EE \text{ [bits/Joule]} = \frac{\text{Secret key rate [bit/s]}}{\text{Power [Joule/s]}} \quad (1)$$

They compared the energy used for different QKD protocols, with both discrete and continuous variables, and observed that, without the post-processing, the continuous variable based protocols uses less energy than those based on discrete variables. However, when taking into account the DSP, the CV based protocols cost more for long distance where channel noises are high, highlighting its importance of the post-processing.

2 Overview of a quantum key distribution protocol

Quantum key distribution protocols have multiple steps. We give a brief overview of a QKD protocol, in our case Gaussian modulated coherent state CV-QKD protocol. Figure 1 gives a visual overview of it.

Prepare Alice prepare states to be sent to Bob. In the case of a continuous variable, she generates $2N$ random numbers x_j in $\mathbb{R}$ according to a Gaussian distribution of mean 0 and variance σ^2 , then generate the states $|\alpha_i\rangle = x_{2i} + x_{2i+1}I$ where I represent the imaginary unit. The state is then transmitted using a laser pulse.

Measure Bob perform measurement on the state sent by Alice. In our case, it is a heterodyne, using a beam splitter and measurement on the two outputs to recover y_{2i} and y_{2i+1}

Information reconciliation Alice and Bob tries to have the same raw key. This step is part of the focus of this work and more details are given later in Subsection 2.2.

Verification After performing the information reconciliation step, Alice and Bob verify they have the same raw key. If they don't have the same raw key, the protocol is aborted. One of them compute the hash of the raw key, and send it to the other.

Parameter estimation Using the pessimistic assumption that all noises are due to Eve’s measurement, Alice and Bob compute a bound on the amount of information Eve could have by comparing their data. Again, if they consider Eve knows too much information, the protocol is aborted.

Privacy amplification After all of these steps, they have raw bits with low entropy and an idea of the amount of information Eve knows. Using this, they can use a construction, a randomness extractor, to transform those bits of weak randomness into fewer bits that are closer to a uniform distribution and such that Eve knows.

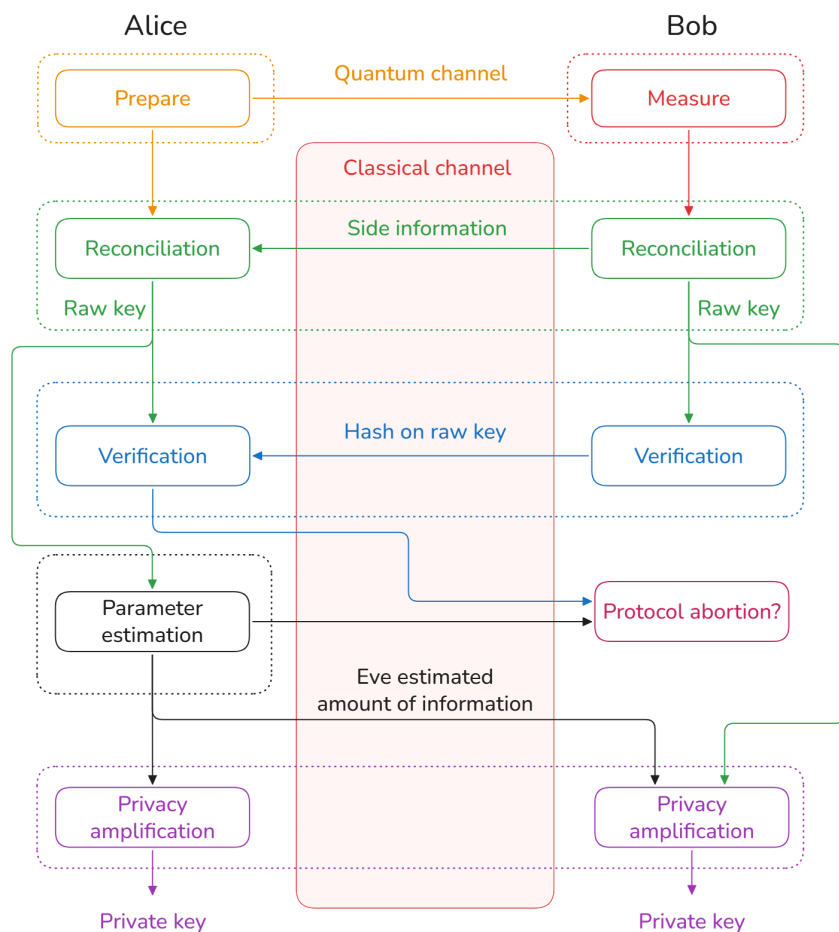


Figure 1: Visual representation of the protocol.
Taken from [Mar+26].

2.1 Privacy amplification

When performing a QKD scheme, Alice and Bob cannot use the raw key immediately because Eve knows some bits. First, they need to make sure that, even with the bits of the raw key known by Eve, she knows none of the bits of the final key. One naive way could have been to

throw away the bits known by Eve; however, she would know some bits without being detected, so Alice and Bob cannot know which bits to throw away. Instead, they use a construction named *Randomness extractor*.

Definition 1. A (k, ϵ) seeded randomness extractor is a function

$$\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$$

such that for any (n, k) distribution X , $\text{Ext}(X, U_d)$ is ϵ -close to U_m , where U_m denotes the uniform distribution over $\{0, 1\}^m$.

Ext is said to be (k, ϵ) -strong seeded randomness extractor if $U_d | \text{Ext}(X, U_d)$ is ϵ -close to U_{m+d} .

In a less formal way, a randomness extractor is a function that, given a seed uniformly generated and some raw bits, returns a new string close to a string generated uniformly. The strong extractor part means that the output is completely decorrelated from the seed. In our case, the seed is communicated on the classical channel and is hence known by Eve.

When choosing the size of the final key, we make the pessimistic assumption that all the noise of the quantum channel is due to measurements done by Eve.

In the following, we will consider two constructions: the Toeplitz extractor and the Trevisan extractor.

The first one is a matrix-vector product between a Toeplitz matrix and the input data as a vector, with $\mathbb{F}_2$ as the base field. A Toeplitz matrix T is an $m \times n$ matrix such that $t_{i+k, j+k} = t_{i, j}$ for all k . This operation can be performed with a polynomial multiplication, using for example a number theoretic transform, and keeping only some terms, with a complexity in $O((m+n) \log(m+n))$.

The second construction we consider is Trevisan's one. This has two components: a one-bit extractor, which compute one bit of the output using the original input and a small seed ; and a weak design, to select the bits of the seed and construct the small seeds using an index. Figure 2 gives a visual representation of this construction.

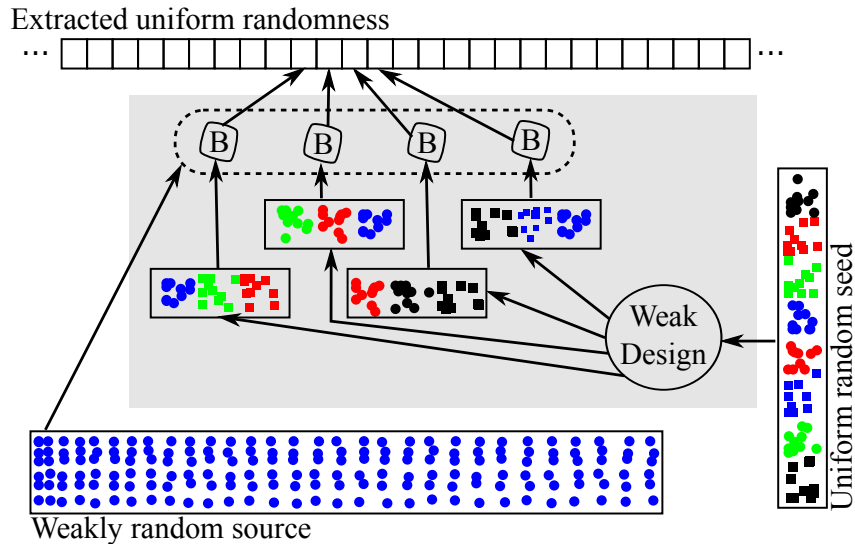


Figure 2: Representation of Trevisan's construction
Taken from [MPS12].

The weak design we used is the block weak design, which uses another weak design, the construction of Hartman and Raz. The construction of Hartman and Raz is based on polynomial evaluation over a finite field, here $\mathbb{F}_{2^t}$, for each of the small seed's bits. For the i -th bit of the key,

the block weak design start by inferring j and k from i , in Cryptomite using a binary search on precomputed values. The k is given to the other weak design, which output its indexes. Those indexes are then reused to compute the new indexes of the bits used for the small seed, by adding $j \cdot t^2$.

For the one-bit extractor, we used the Reed-Solomon Hadamard construction. The first step is the Reed-Solomon step, where the input is interpreted as a polynomial over $\mathbb{F}_{2^t}$, evaluated at a value corresponding to the first part of the small seed. The second step is the Hadamard step, which is a scalar product of the previous result and the second part of the small seed.

2.2 Information reconciliation

The other part of the protocol we experiment is the information reconciliation step. The objective of this part is to make sure both Alice and Bob have the same bits, despite channel and Eve's noises. The idea is that one of the parties reveal on the classical channel some information, allowing the other to get the same result. The efficiency of this step is given by $\beta \leq 1$.

For this part, we used work from [Mar+26]. The implementation we tested use a construction named the virtual channel, which allows the quantum channel to be used as binary input additive white Gaussian noise channel. With this channel, Alice and Bob can use adapted error-correction techniques. In our tests, we use the coset encoding. With this coding, the efficiency is:

$$\beta = R/I_{AB}$$

where R is the code rate, the ratio of the number of bits divided by the size of a codeword ; and I_{AB} the mutual information between Alice and Bob.

2.3 Secret key rate

One important metric is the secret key rate (SKR). It is given by:

$$SKR = \beta I_{AB} - \chi$$

where

- β is the reconciliation efficiency ;
- I_{AB} is the mutual information between Alice and Bob ;
- χ is the Holevo bound, which is a bound on the number of bits Eve knows.

Among the parameters to compute I_{AB} is the channel attenuation T , which decrease with the distance. The other, which we will consider constant, include ξ , V_A and V_{el} which are respectively the excess noise from the channel, Alice's modulation variance and the electronic noises from the receiver ; and η which is the efficiency of the receiver noise.

In practical QKD scheme, we can't send an infinite amount of information. Instead, the data is split in blocks. This impact the previous formula, by adding another term, $\Delta(\epsilon, N)$, and changing the Holevo bound to depend on a security parameter ϵ_{PE} and N_{PE} , the number of bits used for parameter estimation:

$$SKR_N = \beta I_{AB} - \chi(\epsilon_{PE}, N_{PE}) - \Delta(\epsilon, N)$$

Wit this, we can have the evolution of the SKR with respect to the distance. Figure 3 shows this evolution for block sizes, and asymptotically (with infinite size).

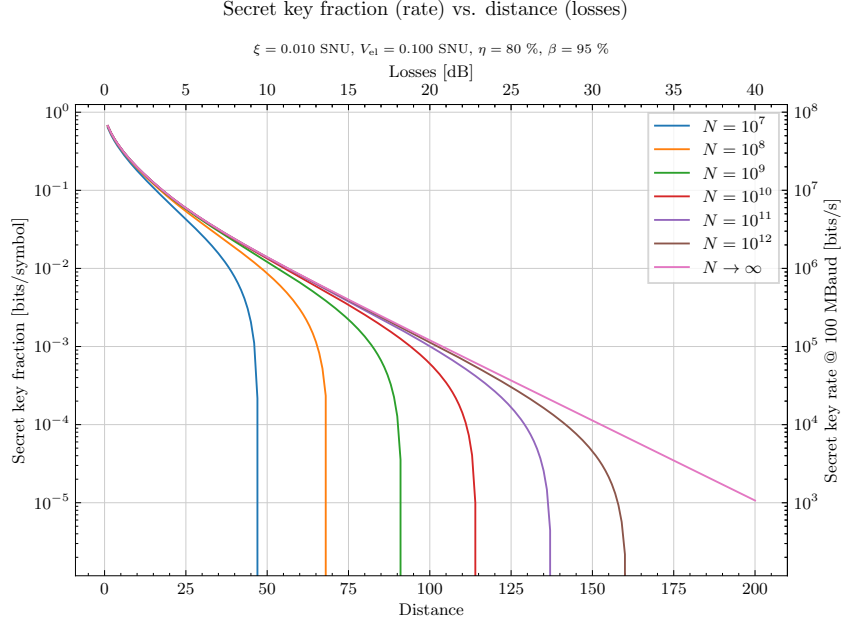


Figure 3: Evolution of the secret key rate with distance.

2.4 Libraries used

The implementations we tested come from two softwares.

Cryptomite [For+25] is a Python library that implements different randomness extractors, used for privacy amplification. Despite being a Python library, the actual implementation is in C++, allowing us to call them from C/C++, without the overhead from the Python environment.

For information reconciliation, we used the HDirac simulator, which accompanied [Mar+26], and implement the protocols discussed. This includes the physical layer, including noises from the quantum channel. The simulator is written in C++ and uses AFF3CT [Cas+19], also written in C++, and can be run with pipeline parallelism. The program is split in different stages, with each stage except the first and the last receiving data from the previous stage and sending the output data to the next one.

We also used OpenMP and Sycl to test parallelism for the privacy amplification. OpenMP is an API for shared-memory multiprocessing. We used it in C++ to test multi-threading implementations. Sycl is a domain specific embedded language based on C++ to program on hardware accelerator, in our case GPUs.

3 Experiments

3.1 Platform

The experiments are done on the Dalek cluster [CAB25]. The cluster is composed of various consumer-grade CPU, typically found in mini-PC or laptops, and gaming GPU. The architecture of the nodes used for the benchmarks are given by the Table 1.

Each of those nodes are equipped with the NCM (Node-Conso Modular) energy measurement mainboards and their probes. The probes are placed on different connectors, which allows us to have a better idea of the power consumption of the different PC components. The voltage and the current are measured by the different probes:

- The p-ATX for the 4 rails Advanced Technology eXtended (ATX) 24-pins connector and the CPU Entry-level Power Supply 12 V (EPS12V). The ATX connector correspond to

most of the PC components, including the USB, the RAM, the SSD, the chipset and the fans ; and the EPS12V for the CPU.

- The p-PCIe for either up to 4 Peripheral Component Interconnect Express (PCIe) 8-pin connector or the 12 V High PoWeR (12VHPWR) connector. In the nodes used, the GPU uses those connectors.
- The p-small for either the USB Power Delivery or the 19 V coaxial connector. This connector is only used to power `iml-ia770` mini-PCs in our case.

The mainboard aggregate the data collected by the probes and communicate them to the node itself, including the energy used from the beginning of the measurement.

The said mainboard is controlled from the node with a utility. The data is printed, either for a given duration or until interrupted, and includes the voltage and the current, which allows us to compute the power, but also the energy used from the beginning of the measurement.

3.2 Methodology

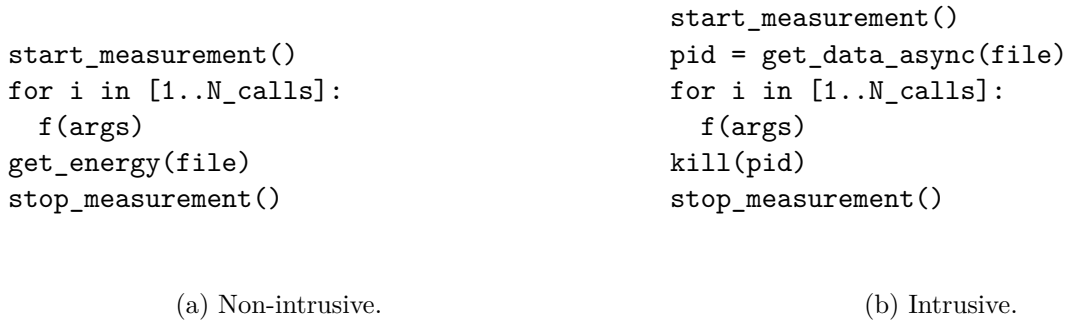


Figure 4: Comparison of the two experiments.

To measure the energy used by a function f with some arguments $args$, we consider two methods, summarized in Figure 4. In both experiments, we measure N_{calls} calls to have a duration longer in comparison to the calls to the utility, and redirect the utility’s output in a temporary file.

The first experiment is the non-intrusive one (Figure 4a), which will measure the energy and the time. It will launch the measurement, calling a certain amount of time the function, then get the energy used during the whole measurement. In the example, `get_energy` hide call to the utility and the redirection.

The second one is the intrusive experiment (Figure 4b), that measure the power during the execution. In this method, another process is created to copy the results to a temporary file in RAM. This method gives more information, but add some contribution due to having another process recording the power consumption, using another core. Again, `get_power_async` hide some complexity.

We also measure the duration, allowing us to compute the missing value with the relation $E = P \cdot \Delta t$, with E the energy, P the power and Δt the duration.

Node(s)	CPU	GPU
az4-n4090-*	AMD Ryzen 9 7945HX	Nvidia GeForce RTX 4090
az4-a7900-*	AMD Ryzen 9 7945HX	AMD Radeon RX 7900 XTX
iml-ia770-3	Intel Core Ultra 9 185H	Intel Arc A770
demo	Apple Silicon (M1 Ultra)	2 Intel Arc Pro B60

Table 1: Nodes used.

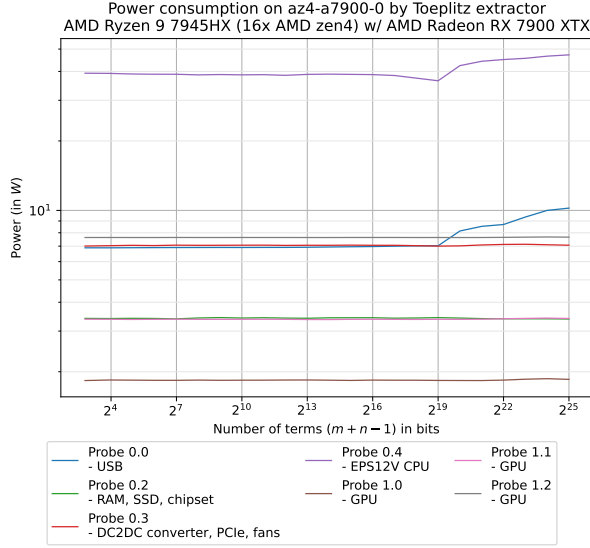


Figure 5: Average power of Toeplitz extractor.

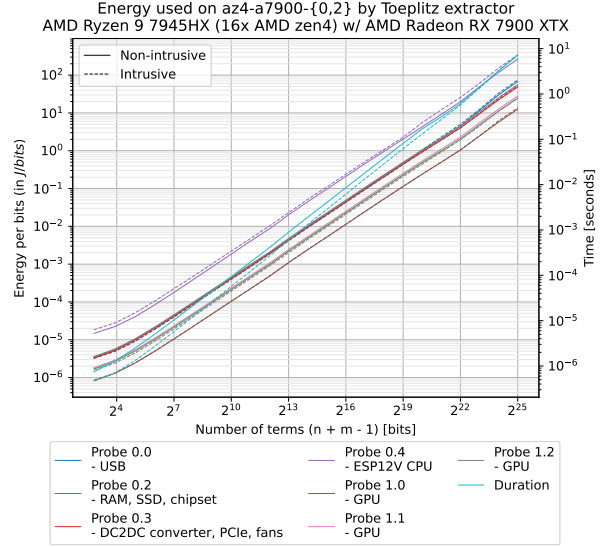


Figure 6: Energy used and duration of a Toeplitz extractor.

4 Results

4.1 First experiments

We started with some experiments to test NCM. Those were done with Cryptomite’s implementation of the Toeplitz extractor. Because it is implemented with a polynomial multiplication, we considered only the number of terms of the polynomials representing the Toeplitz matrix. We first observe that the power is constant for most of the input sizes and only increases toward the end, at sizes 2^{20} and 2^{23} , as shown on Figure 5. The increase is for probes 0.0 and 0.4, which corresponds to the USB for the first one, and the CPU for the second one, increases.

From this result, we can consider that the energy used will mostly follow the duration of the computation. Figure 6 depict the evolution of the energy used and the duration in the same conditions as previously. They both follows an $d \log(d)$ tendency, for d the degree of the polynomial, which corresponds to the complexity of the NTT algorithm.

4.2 Privacy amplification

The second experiment we have done is testing the different privacy amplification algorithms with parameters closer to those found in a QKD context. To do so, we used the results from Figure 7 for the output sizes m . For the input size n , we took $n = \frac{N}{2}$ where N is the number of symbols in a block.

In the case of Toeplitz, we can compute the number of terms of the polynomials manipulated. Because $m \in \llbracket 1, n - 1 \rrbracket$, the sizes of the polynomials for a fixed n can be bounded:

$$n \leq m + n - 1 < 2n$$

With this, we can expect a duration almost constant for a given block size, independently of m . This also correspond to the size of the seed. Figure 8 show the evolution of this value with the distance, for different block sizes, where we clearly see the lower bound for each block sizes.

Using those parameters, we can measure the cost of the Toeplitz extractor with respect of the distance. Figure 9a show the evolution of the throughput with respect to distance and Figure 9b show the same for energy efficiency. We can observe that the energy efficiency follows

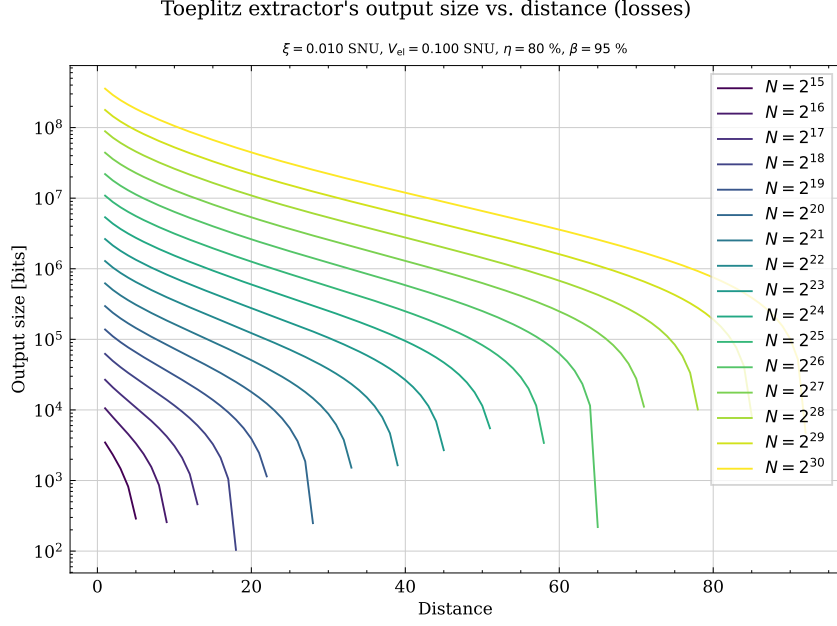


Figure 7: Evolution of Toeplitz extractor's output size with the distance.

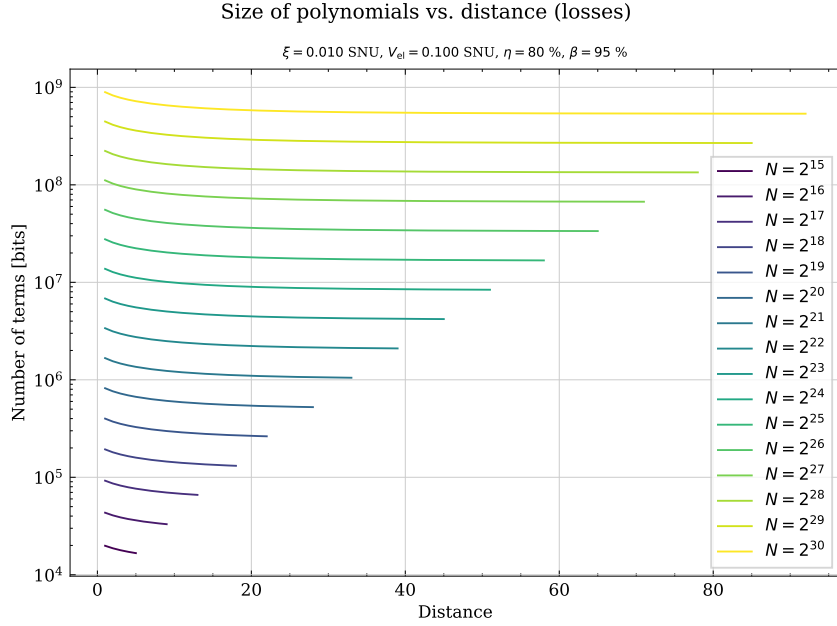
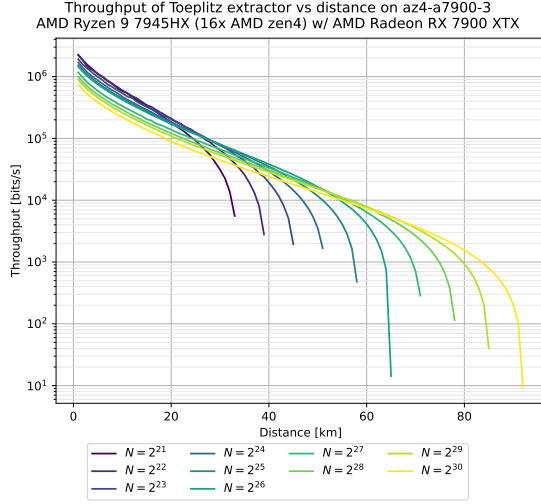


Figure 8: Evolution of the number of terms of Toeplitz's polynomial with the distance.

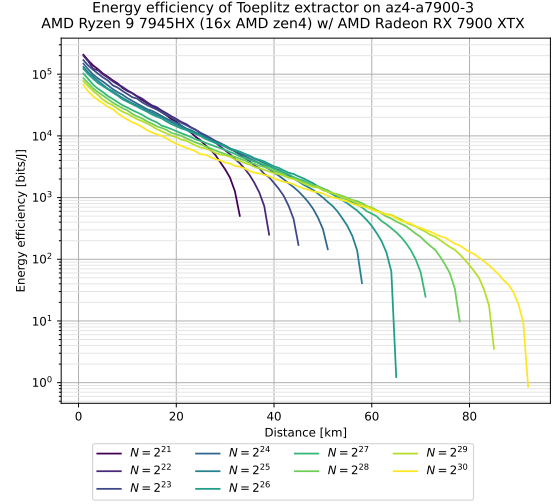
the throughput. This can be explained with Equation 1: as seen in Figure 5, the power is almost constant, hence the energy efficiency is the throughput with a factor. For a distance of 1 km, we have a throughput of around 1 Mbits/s and an energy efficiency of around 0.1 Mbit/J.

Next, we tested the Trevisan extractor. Due to the duration of one call for sizes corresponding to QKD settings, we tested parallelized versions of it, one with OpenMP and on GPU. As explained in Section 2.1, this extractor has a main loop for each bit of the output, which is where we parallelized. Figure 10 shows the results of the OpenMP version with 32 threads.

Both versions have throughput and energy efficiencies that are almost constant, except toward the end where they diverge. On the OpenMP version, we observed an energy efficiency increases with the distance, which happen when m decrease. For the GPU version, it decreases

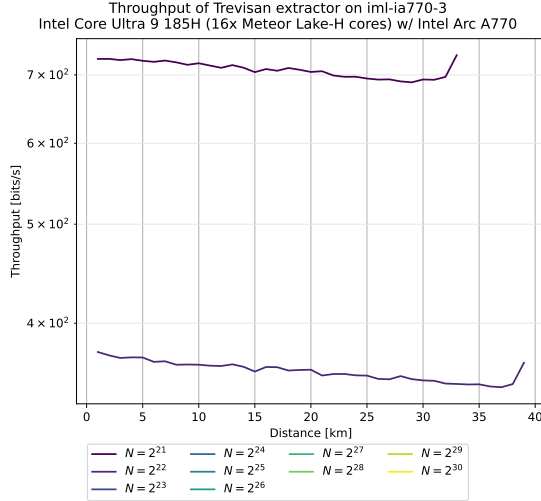


(a) Throughput

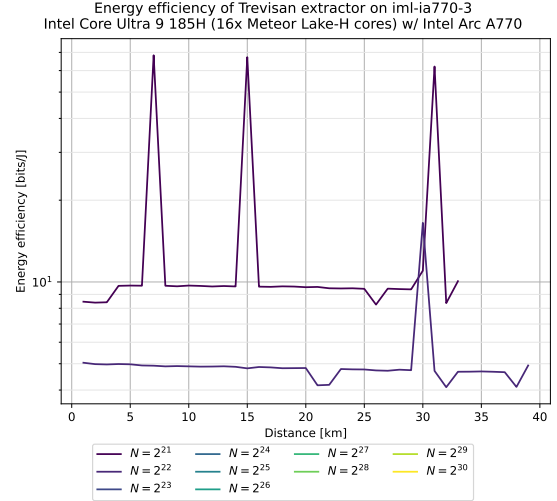


(b) Energy efficiency

Figure 9: Evolution of different metrics with Toeplitz extractor with respect to distance for different block sizes.



(a) Throughput



(b) Energy efficiency

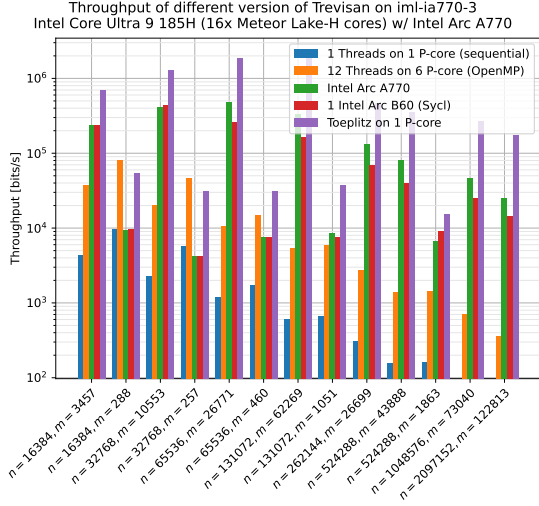
Figure 10: Evolution of different metrics with Toeplitz extractor with respect to distance for different block sizes.

because m becomes too small to use all the cores. In terms of order of magnitude, the GPU version has a better throughput than the CPU version, which was expected, but also a better energy efficiency.

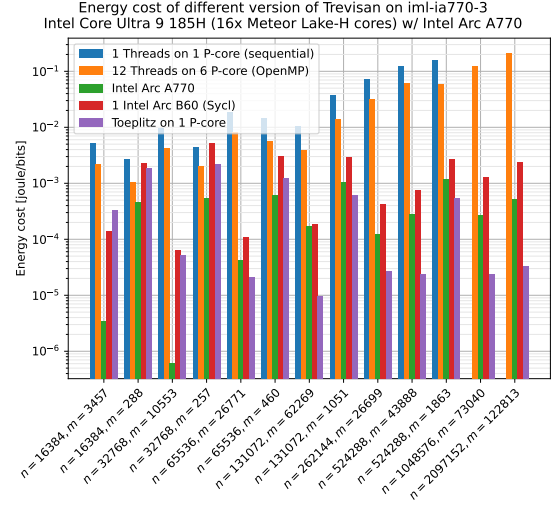
To finish this part, we can compare both extractors with Figure 11. For both metrics, the Toeplitz extractor is better than the Trevisan extractor, even when considering the GPU. However, the advantage of the Trevisan extractor is the size of its seed, which is in $O(\log(\log n))$, whereas the size of the Toeplitz one is $n + m - 1$.

4.3 Information reconciliation

The last part of this internship focused on information reconciliation. We used implementations from HDirac and added calls to NCM at the beginning and the end of each simulation. We tested with the coset coding, for β values ranging from 0% to 100% with a step of 1%. In the



(a) Throughput



(b) Energy efficiency

Figure 11: Comparison of the different extractors.

simulation, the code rate stayed constant at 0.1, so increasing β mean decreasing I_{AB} .

We tested both in sequential (without parallelism) and in pipeline (with parallelism). Figure 12 show the throughput of both versions.

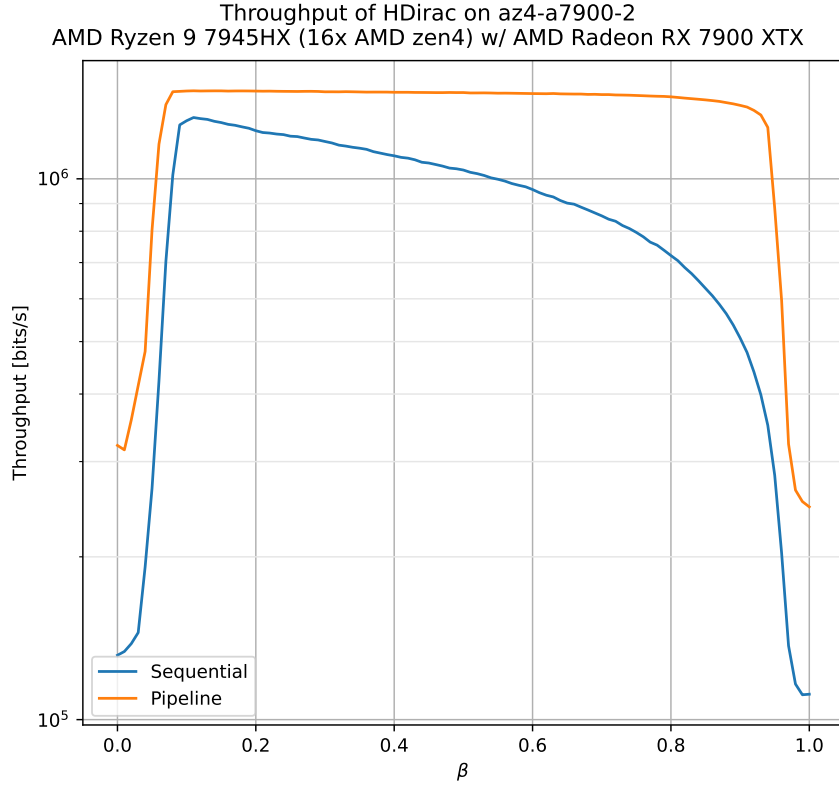


Figure 12: Throughputs of HDirac using coset coding.

As expected, the pipeline has a better throughput than in sequential. The two components which take the most time are the mapping of the virtual channel, which is almost constant, and the decoder, which increase. In sequential mode, the duration of the decoder directly impact the throughput, because the time taken for the whole process cumulate those two parts. In pipeline mode however, those two are on different stages, thus the duration is impacted only by

the most expensive of those two.

For β between 7% and 94% is almost constant around 1.4 Mbits/s for the pipeline, and from 1.3 Mbits/s at $\beta = 11\%$ and decrease with 0.35 Mbits/s at $\beta = 94\%$ for the sequential. For β smaller than 6% of greater than 95%, the frame error increase quickly. In this case, the decoder can't recover the block and will be dropped after verification, reducing the throughput.

We also measured the energy cost, Figure 13 show the energy cost in the same condition.

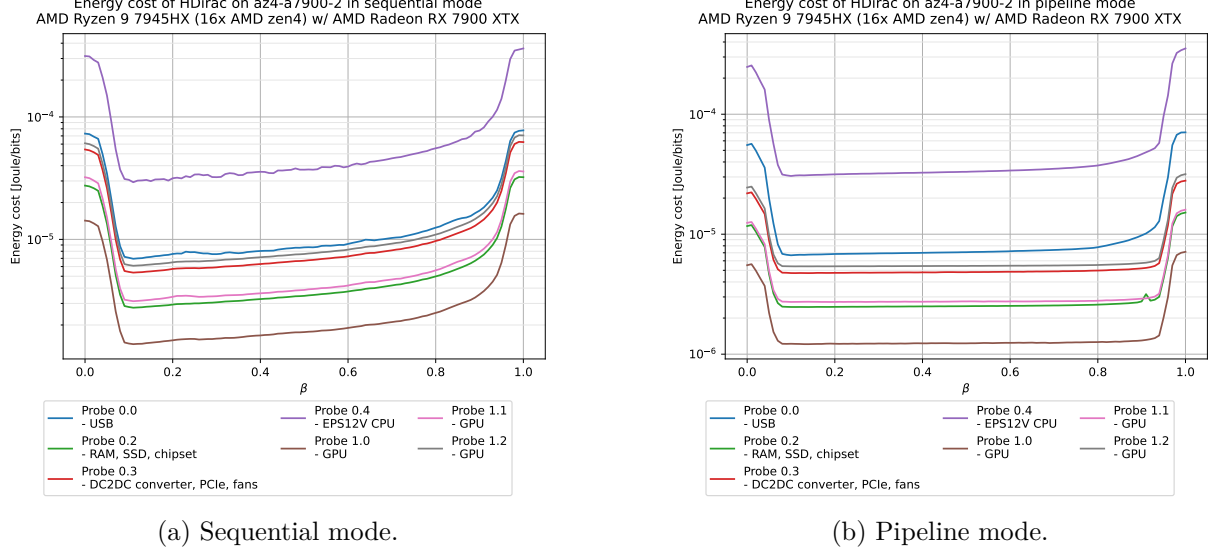


Figure 13: Energy cost of HDirac with coset coding.

The energy cost follows the same tendency as the inverse of the throughput. We have an energy cost from 54 to 90 J/Mbits for β from 8% to 94%, giving an energy efficiency from 11 kbits/J to 18 kbits/J in pipeline mode.

5 Conclusion

In this work, we tested implementations for both privacy amplification and information reconciliation. The Table 2 summarized our results.

	Privacy amplification ($n = 524288$, $m = 43888$)			Information reconciliation	
	Toeplitz	Trevisan (CPU)	Trevisan (GPU)	Sequence	Pipeline
Throughput	354 kbits/s	156 bits/s	81 kbits/s	1.3 Mbits/s	1.4 Mbits/s
Energy efficiency	42 kbits/J	8 bits/J	3.6 kbits/J	< 17 kbits/J	11 – 18 kbits/J

Table 2: Throughputs and energy efficiencies for the different components.

We observed gains for both throughput and energy efficiency when using a GPU, making its use interesting at least for some cases, in our case with Trevisan's extractor when the output size is large enough.

When comparing the throughputs, the limiting factor might seem to be privacy amplification. However, this is due to the fact that this step compress the raw key. When looking to the amount of bits consumed, it is higher than the throughput of information reconciliation, in the case of Toeplitz extractor.

In [Yeh+24], they used block of size 1 Gbits. They considered two values for the cost of the DSP: $\tau_{DSP} = 0.018$ J/symbols, which they measured, and 0.006 J/symbols, they assumed attainable with better implementation. With 0.018, at 80 km, they ended up with an energy

cost of around 10^3 MJ, so an energy cost of 1 J/bits ; and around $2.5 \cdot 10^2$ MJ, so 4 J/bits with 0.006.

In our case, we only have result with such block sizes for the Toeplitz extractor. For the parameters corresponding to a block size of 1 Gbits, the energy used is around $3 \cdot 10^4$ J, so $3 \cdot 10^{-2}$ MJ, which is way lower than the communication and the DSP part.

5.1 Perspective

In this work, we adapted the Trevisan's extractor for a GPU, however we didn't implement a GPU version of Toeplitz extractor, despite the gains for both throughput and increased the energy efficiency. For information reconciliation, we only tested one coding scheme, and didn't try optimization on it, nor porting it on GPU. Finding situations where an implementation is better on CPU or GPU can be interesting, but can become very specific depending on the architecture and algorithms chose.

Another interesting thing is testing and measuring the whole protocol. In this work, we focused on privacy amplification and information reconciliation steps and tested them separately, but verifying if our observations stays when all the components are put together is an open question.

6 Bibliography

References

- [Auf22] Alexia Auffèves. “Quantum Technologies Need a Quantum Energy Initiative”. In: *PRX Quantum* 3.2 (2022). ISSN: 2691-3399. DOI: [10.1103/prxquantum.3.020101](https://doi.org/10.1103/prxquantum.3.020101). URL: <http://dx.doi.org/10.1103/PRXQuantum.3.020101>.
- [CAB25] Adrien Cassagne, Noé Amiot, and Manuel Bouyer. “Dalek: An Unconventional and Energy-Aware Heterogeneous Cluster”. working paper or preprint. Aug. 2025. DOI: [10.48550/arXiv.2508.10481](https://arxiv.org/abs/10.48550/arXiv.2508.10481). URL: <https://hal.science/hal-05186288>.
- [Cas+19] A. Cassagne et al. “AFF3CT: A Fast Forward Error Correction Toolbox!” In: *Elsevier SoftwareX* 10 (Oct. 2019), p. 100345. ISSN: 2352-7110. DOI: <https://doi.org/10.1016/j.softx.2019.100345>. URL: <http://www.sciencedirect.com/science/article/pii/S2352711019300457>.
- [For+25] Cameron Foreman et al. “Cryptomite: A versatile and user-friendly library of randomness extractors”. In: *Quantum* 9 (Jan. 2025), p. 1584. ISSN: 2521-327X. DOI: [10.22331/q-2025-01-08-1584](https://doi.org/10.22331/q-2025-01-08-1584). URL: <https://quantum-journal.org/papers/q-2025-01-08-1584/>.
- [Mar+26] Lucien Martial et al. *Multidimensional Reconciliation in Continuous-Variable QKD: Review, Coding Schemes, and Open Source Simulation*. 2026. DOI: [10.48550/ARXIV.2606.02323](https://arxiv.org/abs/10.48550/ARXIV.2606.02323). arXiv: [2606.02323](https://arxiv.org/abs/2606.02323) [cs.IT]. URL: <https://arxiv.org/abs/2606.02323>.
- [MPS12] Wolfgang Maurer, Christopher Portmann, and Volkher B. Scholz. *A modular framework for randomness extraction based on Trevisan’s construction*. 2012. DOI: [10.48550/ARXIV.1212.0520](https://arxiv.org/abs/10.48550/ARXIV.1212.0520). arXiv: [1212.0520](https://arxiv.org/abs/1212.0520) [cs.IT]. URL: <https://arxiv.org/abs/1212.0520>.
- [Yeh+24] Raja Yehia et al. “Energetic Analysis of Emerging Quantum Communication Protocols”. working paper or preprint. Oct. 2024. URL: <https://hal.science/hal-04740054>.