Hacking the privacy amplification of quantum key distribution with machine learning, and countermeasures: An argument for considering classical side channels in quantum key distribution

João Diogo Ferreira Bravo
joaofbravo@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2022

Abstract

Quantum key distribution (QKD) exploits the principles of quantum mechanics to generate and distribute private keys using quantum systems and an authenticated public classical channel. Although it offers information-theoretical security, its physical implementations usually do not due to unintended sources of information leakage, called side channels, which eavesdroppers exploit to obtain information about the private key. Side channels in QKD are either similar to those found in classical implementations or related to the quantum processing. Current research and development have often neglected the former. We propose a classical-side-channel attack on the privacy-amplification step of a general QKD protocol based on matrix hashing, using machine-learning techniques to analyse its power-consumption leakage. We analyse multiple simulated scenarios and are often able to recover the full private key. Gradient-boosting machine was the best-performing model in virtually every scenario, recovering the full key for high-enough measuring-instrument sampling rates with any hashing-matrix size and noise level tested. In case of a non-perfect model, we devise a strategy, based on analysing the confusion matrices of the model, which can make a brute-force search for the key feasible. In our attack approaches, we find a trade-off between the computational resources needed and the attack performance. We also discuss countermeasures based on noise insertion, masking and randomization techniques, some of which can restore the information-theoretical security of the QKD protocol. This work demonstrates that machine-learning techniques can be used to robustly characterize the leakages in a QKD implementation and generate powerful attacks.

Keywords: quantum key distribution, privacy amplification, side-channel attack, power consumption, machine learning, gradient boosting

1. Introduction

QKD consists in generating and distributing a private key using quantum systems and an authenticated public classical channel, i.e., a classical channel where an eavesdropper, Eve, is not able to impersonate one of the two parties but can still eavesdrop on them. All QKD protocols use the fact that interacting with a quantum system perturbs its state to detect the presence of Eve. QKD offers information-theoretical or unconditional security [1], i.e., it can be proven secure with information-theory methods without imposing any conditions on the resources available to Eve. Thus, in theory, QKD protocols cannot be broken even by an Eve with unlimited classical and quantum computing power.

Despite its information-theoretical security, QKD faces a few practicality challenges. Some of the major ones concern its limitations in transmission distance and key generation rate [1]. These limitations can be solved with the use of repeaters, also known as nodes or relays, along the communication channels [2, 3].

Another major challenge is due to unintended sources of information leakage in physical implementations [1], known as side channels, since these leakages are often highly correlated with the generated key. An eavesdropper can interact with and exploit physical side channels rather than the abstract protocol to obtain information about the generated key without being detected.

We divide side channels in QKD into two categories according to their origin. We refer to side channels related to the quantum processing of a QKD protocol as quantum side channels and to those also found in classical implementations as classical side channels.

There has been substantial research on quantum-side-channel attacks [1], with specific security patches proposed for each attack. Besides these, new classes of protocols have been proposed which claim to make side-channel attacks fully or partially obsolete. The most-well-known classes are measurement-device-independent (MDI) QKD [4] and device-independent (DI) QKD [5]. The former claims to remove all side channels from detection components, while the latter claims to remove all side channels from all implementation components. However, we argue that DI-QKD protocols may still be exposed to classical side channels since these do not disturb the transmitted quantum systems. Moreover, MDI-QKD protocols may still be exposed to both classical and quantum side channels in source components.

Unfortunately, classical-side-channel attacks are often not taken into account in QKD research and development. In fact, only very recently there have been a few proposals of such attacks on QKD implementations, targeting side channels such as timing information [6], power consumption [7, 8], emanated electromagnetic radiation [9, 10] and cache state [11] of specific implementation devices. However, most devices used in a QKD implementation may be vulnerable to classical-side-channel attacks. Moreover, QKD networks with multiple remotely-located repeaters can be particularly exposed to such attacks since constant surveillance of all repeaters can be difficult.

In this work, we study the exploitability of classical side channels in QKD implementations with the aid of machine-learning techniques. We propose and analyse a classical-side-channel attack on the privacy-amplification step of a general QKD protocol based on matrix hashing, using its power-consumption leakage. For this, we simulate the power-consumption leakage of a specific implementation for privacy amplification in multiple attack scenarios. Then, we use two approaches to analyse all leakages with machine-learning techniques and assess the best approach and machine-learning model to perform our proposed attack. Due to the success of our attack, we propose countermeasures to prevent similar side-channel attacks and discuss their efficiency.

A comprehensive review of the main quantum- and classical-side-channel attacks demonstrated in QKD implementations and their respective countermeasures can be found in [12].

2. Background

2.1. Hash functions

A hash function is any function $H : X \to K$ such that $|X| \geq |K|$ [13], where $|X|$ is the cardinality of X. The objects in K are called hashes.

Let $\mathcal{H} = \{h : X \to K\}$ be a family of hash functions. $\mathcal{H}$ is universal$_2$ if [13]

$$x \neq x' \implies P\big(H(x) = H(x')\big) \leq \frac{1}{|K|}, \quad \forall\, x, x' \in X, \tag{1}$$

when H is chosen uniformly at random from $\mathcal{H}$.

2.2. Quantum-key-distribution protocols

The most well-known QKD protocols can be divided into two categories, depending on which quantum property they exploit [1]: entanglement-based (EB) protocols take advantage of quantum entanglement and prepare-and-measure (PM) protocols exploit of the probabilistic nature of quantum systems, usually referred to as quantum uncertainty or quantum indeterminacy, to detect Eve. Since maintaining a shared entangled state at long distances is still very challenging in practice, EB-QKD protocols are generally considered idealized versions of PM-QKD protocols [14]. These two approaches can be further divided into two families of protocols depending on the nature of the quantum systems used: discrete-variable protocols and continuous-variable protocols.

Currently, a typical QKD protocol consists of a quantum-processing stage followed by a classical-postprocessing stage [14]. In the former, the communicating parties, Alice and Bob, generate, share and measure quantum systems to produce partially-shared strings while, in the latter, they: sift their strings to generate a raw key and to detect any intrusion by Eve; correct differences in their raw keys to obtain a shared identical corrected key; and extract a secret key from their corrected key which is independent of any knowledge Eve may have acquired during the protocol. At the end of the protocol, the resulting secret key can be used to encrypt messages over an untrusted classical channel, i.e., a classical channel which may be subject to eavesdropping and does not require authentication.

The ratio of the key length to the total number of shared quantum systems is designated by key generation rate.

In [12], we present a brief review of some of the main theoretical and experimental progress in QKD, namely the main breakthroughs in security-analysis techniques, a description of the main components of QKD implementations, the main algorithms for classical postprocessing and the use of classical and quantum repeaters for extending the transmission distance in QKD.

2.3. Privacy amplification

Up to its last step, a QKD protocol necessarily leaks information about the corrected key to Eve. Therefore, privacy amplification is required to distill a secret key from the partially-exposed corrected key [15]. Classical privacy amplification is performed over the authenticated public channel.

In [15], they proposed the use of universal$_2$ families of hash functions as a secure way to perform privacy amplification. To do so in a QKD protocol, Alice and Bob first estimate an upper bound on the side information that Eve may have on the corrected key. Let the corrected key have n bits and that upper bound be $n - l$ bits. Then, they randomly choose a hash function H_{pa} from a universal$_2$ family which extracts l secret bits from n bits and share it over the authenticated public channel. Then, by applying the hash function on their corrected keys, $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, they will get identical secret hashes with l bits, $\boldsymbol{k}_A = H_{pa}(\boldsymbol{x}_A)$ and $\boldsymbol{k}_B = H_{pa}(\boldsymbol{x}_B)$, with high probability. These become the final secret key.

In general, linear privacy amplification, such as matrix hashing, is commonly used in QKD due to its low computational complexity [16]. Let $H_{\boldsymbol{M}}(\boldsymbol{x}) = \boldsymbol{M} \cdot \boldsymbol{x} \mod 2$ be the logical multiplication of a matrix $\boldsymbol{M} \in \{0,1\}^l \times \{0,1\}^n$ and a vector $\boldsymbol{x} \in \{0,1\}^n$. The modulo refers to the value of each entry of the resulting vector. Then, the set $\{H_{\boldsymbol{M}}\}_{\boldsymbol{M}}$ is a universal$_2$ family of hash functions [13]. However, a random seed to choose such a function must have $l \times n$ bits. In QKD, the most-frequently-used universal$_2$ hash functions are based on binary Toeplitz matrices [17], which require much smaller seeds.

A $l \times n$ Toeplitz matrix has entries which are constant along each of its diagonals as

$$\boldsymbol{T} = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & & \\ a_2 & a_1 & a_0 & & \vdots \\ \vdots & & & \ddots & \\ a_{l-1} & & \cdots & & a_{l-n} \end{bmatrix} . \qquad (2)$$

It can also be written concisely as $T_{i,j} = a_{i-j}$, where $T_{i,j}$ is the entry (i, j) of $\boldsymbol{T}$. Such a matrix can be parametrized by a seed with only $l + n - 1$ values.

More recently, [18] showed that the concatenation of a Toeplitz matrix with an identity matrix, commonly referred to as modified Toeplitz matrix, also forms a universal$_2$ family of hash functions. With this family, we only have to generate a $l \times (n-l)$ binary Toeplitz matrix $\boldsymbol{T}$ and concatenate it with a $l \times l$ identity matrix $\mathbb{I}$ as $(\boldsymbol{T}, \mathbb{I})$, for instance. Thus, we only need to generate a random seed with $n - 1$ bits to choose a modified Toeplitz matrix.

2.4. Side-channel attacks

Since side channels are directly related to the devices used in implementations, side-channel attacks are generally implementation specific. Side-channel attacks can be divided into two phases [19]: an optional preparation phase in which Eve profiles and characterizes the leakages with a possibly-simulated device similar to the one on which she intends to perform the attack; and the exploitation phase in which she performs the attack on the real target device with the intent of extracting information about the secret key.

2.5. Machine learning

Machine learning is a subfield of artificial intelligence which provides an automated way of doing data analysis. It encompasses a set of methods which automatically detect patterns in data and use them to predict new data or make decisions involving uncertainty [20].

2.5.1. Supervised learning

In our analysis, we will deal with a supervised-learning problem [20], namely a multi-label classification, in which the models receive a dataset with inputs and their desired binary outputs and their goal is to learn a mapping from inputs to outputs. This dataset consists of N samples, or records, each with k input features and l binary target features. It is represented by a $N \times (k + l)$ matrix.

In a typical supervised-learning analysis, we start by gathering the data and building the full dataset. Then, we split that dataset into a training and a testing dataset. We preprocess each dataset separately to prevent any information leakage due to using statistics which depend on data from the other dataset. This leakage can introduce a bias in the analysis, possibly leading to an overestimation of the model's general performance [20]. The preprocessing usually consists in removing inconsistencies and redundant or irrelevant features, and applying transformations such as scaling the data. In our analysis, we tested multiple transformations, namely scaling all features individually to the range $[0, 1]$, standardizing all features individually to mean 0 and variance 1, scaling all features individually to median 0 and range between the first and the third quartiles equal to 1, and performing no scaling.

Then, we choose the models which we want to assess. We assessed the logistic regression [20], the k-nearest neighbours (kNN) [21], multiple variations of naïve Bayes [20], the support-vector machine (SVM) [22], the decision tree [20], a random forest [23] and a stochastic gradient-boosting machine (GBM) [24] with regression decision trees.

These models are described by two types of parameters [25]. We refer to the variables which are iteratively adjusted during training as parameters and to the variables which we manually adjust before training as hyperparameters. Optimizing the hyperparameters of a model is essential to prevent its under- or overfitting the data [25]. We define a finite set of hyperparameter combinations for each model and perform hyperparameter tuning using the training dataset. The technique we used to perform this tuning was a grid search [26] with 5-fold cross validation [20] and asynchronous successive halving (ASHA) [27]. This consists in testing each hyperparameter combination by dividing the data into 5 subsets and rotatively using 4 of them to train l models, one for each target, and the other subset to assess the combined performance of that ensemble on previously-unseen data, with a chosen metric. We use the average Hamming loss [28] of the ensemble predictions to perform this assessment. It is defined as the average fraction of wrongly-predicted outputs,

$$\overline{\text{Hamming}} = \frac{1}{N \cdot l} \sum_{i=1}^{N} \sum_{j=0}^{l-1} \mathbb{1}\left(\hat{y}_j^i \neq y_j^i\right) , \qquad (3)$$

where $\hat{y}_j^i$ is the predicted value of the j-th output for record i, y_j^i is the corresponding real output and $\mathbb{1}$ is the 0-1 indicator function which outputs 1 if its argument is true and 0 if false. In our analysis, all output variables are equally important, so a lower average Hamming loss indicates a better ensemble. The 5 assessment results are then combined by averaging. The hyperparameter combinations are tested in parallel with the low-performing ones periodically dropped during training to focus the computational resources on the most promising combinations. In the end, we know the best hyperparameter combination for each of the chosen models with high probability, which we use to retrain an ensemble of that model with the full training dataset.

Finally, we evaluate each final ensemble with the testing dataset and the same metric to assess its generalization ability, i.e., its performance when predicting outputs for previously-unseen data.

We can also further assess the quality of each ensemble by analysing the confusion matrix [20] of each of its models. This matrix is built by assigning to the entry (i, j) the number of records with real target i and predicted target j.

In [12], we detail the supervised-learning techniques used in a mostly-self-contained way and provide mathematical descriptions of the models used, along with explanations of their hyperparameters. We also detail the hyperparameter subspaces allowed for each model in our analysis.

3. Side-channel attack on privacy amplification

Our aim is to estimate how many secret-key bits Eve can obtain by intercepting the power-consumption trace of the privacy-amplification step of a general QKD protocol and extracting information from it with machine-learning techniques, instead of the more-commonly-used simple power analysis (SPA) or differential power analysis (DPA) [29].

3.1. Power trace of privacy amplification

At the time of this work, there was no physical hardware, similar to that used in laboratory or commercial QKD implementations, available to us, from which to measure power consumption. Therefore, we simulated the power traces of a specific physical system for privacy amplification with matrix hashing. To do so, we consider what happens at the low hardware level. Firstly, a matrix-vector multiplication with bits can be decomposed into bit multiplications and additions, like in algorithm 1.

Algorithm 1 Logical matrix-vector multiplication.

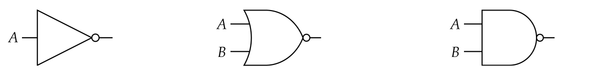
Input: $l \times n$ matrix $\boldsymbol{M}$, n-vector $\boldsymbol{x}$
Output: l-vector $\boldsymbol{k}$

```
1: procedure MATRIX_VECTOR_MULTIPLICATION($\boldsymbol{M}, \boldsymbol{x}$)
2:     for $i \leftarrow 0$ to $l-1$ do
3:         $k_i \leftarrow 0$
4:         for $j \leftarrow 0$ to $n-1$ do
5:             $k_i \leftarrow (k_i + M_{ij} \cdot x_j) \mod 2$
6:         end for
7:     end for
8:     return $\boldsymbol{k}$
9: end procedure
```

We ignore the power consumed by the initialization of the output vector since it is redundant to our analysis. The remaining operations can be further decomposed into the logic gates which are implemented by the hardware. There are many ways to decompose bit additions and multiplications into logic gates. We base our decomposition on the logic gates whose power consumption was analysed by [30], namely the NOT gate or inverter, which takes one binary input and outputs its logical negation, the 2-input NOR gate, which outputs the negation of the logical disjunction of the two binary inputs, and the 2-input NAND gate, which outputs the negation of the logical conjunction of the two binary inputs. These gates are represented by the symbols in figures 1a to 1c, respectively.

The multiplication of two bits, $A \cdot B$, described by an AND gate, can be decomposed into

$$\text{AND}(A, B) = \text{NOT}(\text{NAND}(A, B)), \tag{4}$$



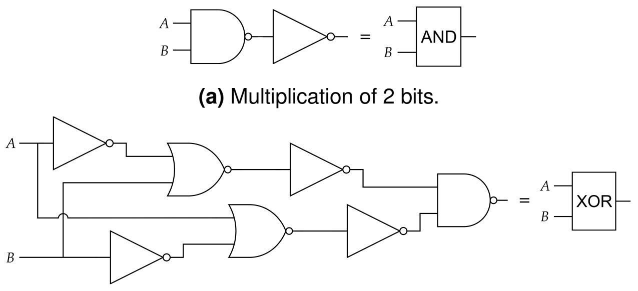
(a) NOT gate. **(b)** 2-input NOR gate. **(c)** 2-input NAND gate.
Figure 1: Symbols for the gates used.

and the addition of two bits, $A + B \mod 2$, described by a XOR gate, can be decomposed into

$$\text{XOR}(A, B) = \text{NAND}(\text{NOT}(\text{NOR}(\text{NOT}(A), B)), \text{NOT}(\text{NOR}(\text{NOT}(B), A))) \tag{5}$$

after some Boolean algebra.

When processing an addition, there are multiple orders in which the individual gates can be processed. The logic diagrams of these multiplication and addition operations are shown in figures 2a and 2b, with the chosen gate-processing order made explicit.



(a) Multiplication of 2 bits.

(b) Addition of 2 bits.

Figure 2: Logic diagrams of the operations used in the simulated implementation for privacy amplification, time-ordered from left to right.

For simplicity, we assume each gate takes the same time to be processed. Then, Eve is able to obtain the power consumption of every gate by measuring the current dissipation at a constant rate, as most devices do, thus acquiring the consumption of a single gate per data point.

The power consumption of these gates varies according to their input and to the size of the transistors which make them up, decreasing as the transistors become smaller [30]. We base the consumption of our simulated implementation on a simplified model of a commercial 0.35-μm-transistor CMOS technology analysed in [30], taking only the dissipated static current into account. This dissipation component gains an increasing importance as the devices become small, specially in the sub-micrometer range, while the other components lose importance [30]. The dissipated static current assumed for the logic gates used is presented in table 1, according to their input.

There are also multiple orders in which the various multiplications and additions of the full logical matrix-vector multiplication can be processed by the hardware. We assume an implementation where the operations are computed row by row, and, for each row, all multiplications are computed before the additions. The block diagram of the implementation chosen to process the logical scalar product of each hashing-matrix row with the corrected key is shown in figure 3, with the operation-processing order made explicit.

Table 1: Static current dissipation of the NOT, 2-input NOR and 2-input NAND gates based on a commercial 0.35-μm-transistor CMOS technology [30].

Gate	Input		Static current (pA)
NOT	0		6.73
	1		7.45
NOR	0	0	13.50
	0	1	12.80
	1	0	7.93
	1	1	5.93
NAND	0	0	5.79
	0	1	7.00
	1	0	11.77
	1	1	14.91

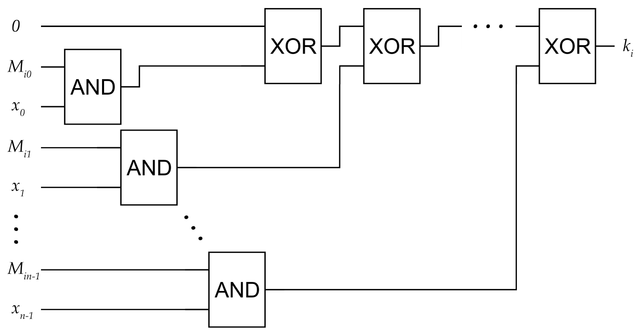


Figure 3: Block diagram of the implementation chosen to process the privacy-amplification operations, namely the logical scalar product of each row i of the hashing matrix $\boldsymbol{M}$ with the corrected key $\boldsymbol{x}$, time-ordered from left to right.

We also note that a regular power trace usually has an overhead contribution from the other processes running on the processor. We assume a constant overhead, which can be removed from the power trace when processing the data. This may be achieved with DPA or other statistical techniques, for instance, and is specially feasible when using simpler circuits dedicated to processing the QKD protocol, such as field-programmable gate arrays (FPGA) or application-specific integrated circuits (ASIC), commonly used in real-world QKD implementations [1].

We also assume noisy-measurement scenarios. There are multiple noise mechanisms in electronic systems [31]. By the central limit theorem, the sum of many statistically-independent random processes tends towards a Gaussian distribution. Thus, it is reasonable to assume most total noise in real-world systems has such a distribution of instantaneous amplitudes with time. We simulate an additive white Gaussian noise [31] of mean zero and standard deviation σ chosen so that the mean absolute value of the noise in all data points ε has appropriate dimensions for our scenarios. The standard deviation may be computed by numerically solving

$$\int_{-\infty}^{\infty} \frac{|x|}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = \varepsilon \qquad (6)$$

with respect to σ.

We also consider measuring instruments with different sampling rates. In order to consider values independent of a time scale, we consider sampling intervals, in gates, instead. For instance, an instrument could be able to capture every gate processed or only one out of every three gates. We assume all samplings start with the first gate processed in the protocol.

We use *Python* to run the simulations. We first randomly generate a $l \times n$ binary modified Toeplitz matrix and 10000 corrected keys with n bits. Then, we run the privacy-amplification procedure with that matrix for each corrected key, generating the current dissipated by the gates processed in the logical matrix-vector multiplication, according to table 1, and a noise component for each data point. We then downsample the data according to the sampling interval s considered, keeping only the first of every s data points. The discarded data points correspond to the processing of the privacy-amplification operations done during the dead time of the measuring instrument.

3.2. Machine-learning analysis

After simulating the measurement of the power traces from the considered hardware implementation, we use machine-learning techniques to extract information about the secret key from them. A record is the subset of data corresponding to one privacy-amplification procedure. We use 80% of the records for training and the other 20% for testing. We use the techniques and follow the approach described in section 2.5.1.

We use the *scikit-learn* machine-learning library for *Python* [28], in which some of the mentioned machine-learning algorithms are implemented. All the code used for this work can be found on GitHub[1], along with comments detailing the nuances of its implementation.

3.3. Attack scenarios and approaches

We assess the robustness of this attack by using it to try to break privacy-amplification protocols with different hashing-matrix sizes, measurement noise levels and instrument's sampling intervals. We simulate scenarios with all combinations of:

- Modified-Toeplitz-matrix size: 8×16, 16×32, 32×64;
- Mean absolute value of measurement noise: 0 pA (no noise), 0.1 pA, 1 pA;
- Instrument's sampling interval: 1, 3, 5, 10, 20 and 30 gates.

The noise levels were chosen to be near the order of magnitude of the current dissipation of the gates considered.

We test two variations of the machine-learning analysis to assess the best approach for Eve. We note that the goal of this analysis is not to empower possible eavesdroppers, but to understand the worst-case outcome for Alice and Bob.

Firstly, we try to find the full secret key by assigning each of its l bits to a different target. We refer to each ensemble of l models used to predict one key as

[1] `https://github.com/joaofbravo/Hacking_QKD_with_ML`

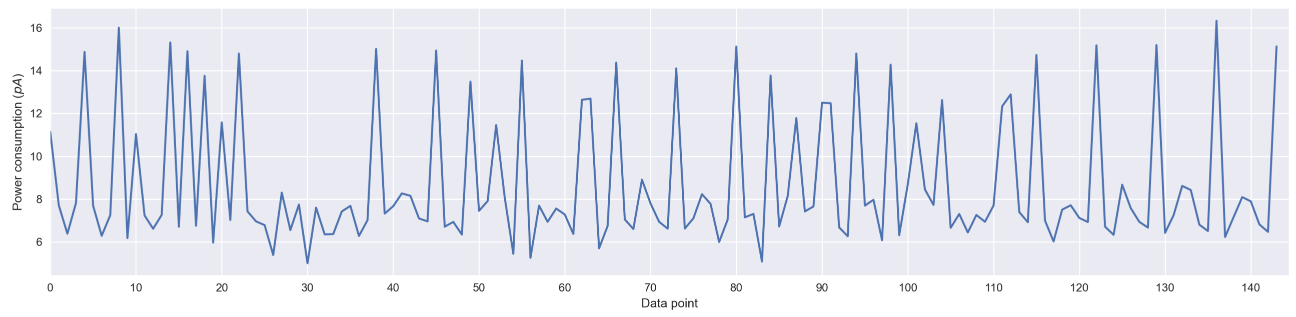


Figure 4: Power trace of the product of the first hashing-matrix row with a corrected key, for a 8×16 matrix, noise level of 1 pA and sampling interval of 1 gate.

full-matrix model, since each individual model of the ensemble receives input data generated from operations involving the full hashing matrix.

To try a less time-consuming approach, we split the dataset rows, which correspond to full power traces, into subrows corresponding to the power trace of the logical scalar product of each matrix row with the corrected key. Instead of 10000 records, we have $10000 \cdot l$ records. Since these logical scalar products fully determine one bit of the secret key, we train models for each of the l secret-key bits with the 10000 subrows which correspond to the operations determining that bit. This approach is less computationally expensive since the models use much less features. By using less irrelevant data, their prediction accuracy may increase. However, scenarios with sampling intervals larger than 1 gate can lead to subrows with one less data point. Since the models need records with the same number of variables, we add a neutral data point of 0 pA at the beginning of those subrows. We refer to each ensemble of l models used to predict one key as row model, since each individual model of the ensemble takes input data generated from operations involving only one row of the hashing matrix.

4. Results and discussion

4.1. Power traces

In figure 4, we present the simulated power trace of the logical scalar product of the first 8×16-matrix row with one corrected key, using a noise level of 1 pA and a sampling interval of 1 gate. The initial current peaks should correspond to the processing of the NAND gates of bit multiplications, with the highest peaks likely corresponding to $1 \cdot 1$. The final higher current peaks should correspond to the last NAND gates processed in each bit addition, while the smaller peaks likely correspond either to NAND or NOR gates. However, the high noise level clearly perturbed some of these operations. In fact, some NAND(1, 1) operations, clearly identifiable in the corresponding scenarios with noise level up to 0.1 pA, now have current dissipation similar to NOR(0, 0) operations, making them indistinguishable.

Lowering the resolution of the measuring instrument also leads to a great loss of information, since Eve collects fewer data points. These information losses affect the performance of the models in each considered scenario.

4.2. Full-matrix models

A summary of all model results from the two taken approaches can be found in [12]. As an example, we present the average Hamming loss of the keys predicted by the full-matrix models, by sampling interval, for a 32×64 hashing matrix and noise level of 0.1 pA, in figure 5.

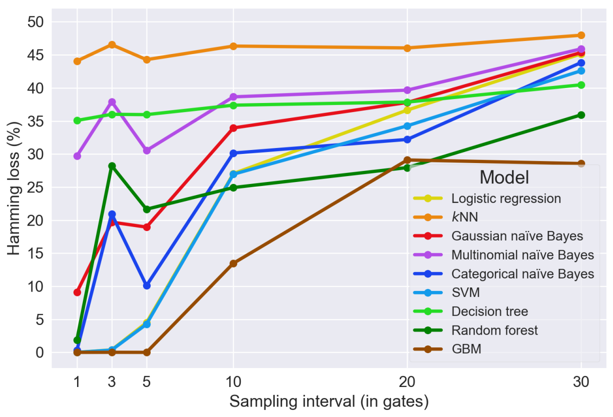


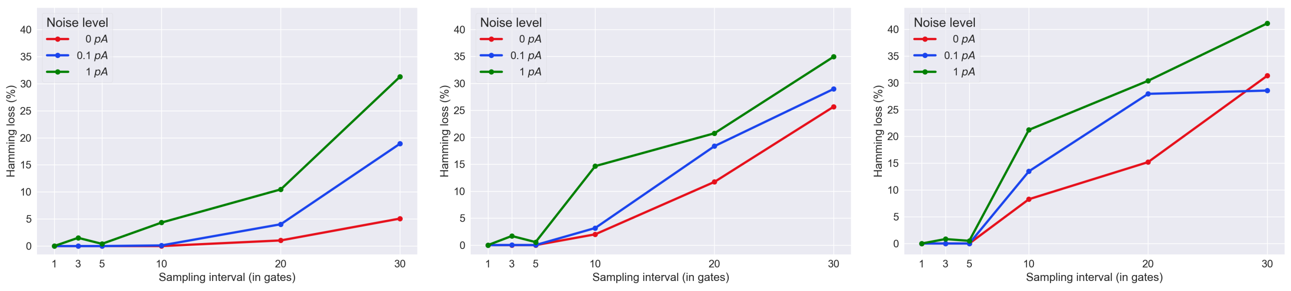
Figure 5: Average Hamming loss of the predicted keys of full-matrix models by sampling interval, with a 32×64 hashing matrix and a noise level of 0.1 pA.

Across all considered scenarios, GBM was the best-performing model, virtually always being able to predict the full secret key for sampling intervals up to 5 gates and all noise levels. For sampling intervals up to 3 gates and noise level up to 0.1 pA, SVM and logistic regression showed a similar performance to GBM. However, GBM was the only model capable of consistently predicting the full secret key in very noisy environments, for sampling intervals larger than 1 gate.

Random forest and the categorical and Gaussian naïve Bayes had a generally-poorer performance, followed by decision tree and the multinomial naïve Bayes, with kNN being the worst. Despite this, some of these models were still able to always predict the full secret key when sampling all gates, for noises up to 0.1 pA.

With imperfect sampling, the quality of the machine-learning models depends on the subset of gates sampled. This explains why some models perform better with a sampling interval of 5 gates than with one of 3 gates.

After performing such a study, Eve then uses the best model according to the particular scenario of the



(a) For a 8×16 hashing matrix. **(b)** For a 16×32 hashing matrix. **(c)** For a 32×64 hashing matrix.

Figure 6: Average Hamming loss of the predicted keys of the best full-matrix model by sampling interval, for each noise level and hashing-matrix size.

attack. We summarize the average hamming loss of the best full-matrix models for each scenario, by sampling interval, in figures 6a to 6c.

There was no significant performance degradation with the increase of the hashing-matrix size for sampling intervals up to 5 gates. In these scenarios, the degradation with the increase in noise level was only significant for a sampling interval of 3 gates. Nevertheless, in those cases, having a noise level of 1 pA only slightly increased the Hamming loss by around 1%. For sampling intervals larger than 5 gates, the information was not nearly enough to predict every bit of the secret key and the performance dropped considerably, with the increases in hashing-matrix size and noise level having great impact in performance.

The success of this attack even in very noisy scenarios can be explained by k_i being fully determined by multiple subsets of operations involving row i, according to algorithm 1. For instance, it can be fully determined by the last addition involving row i, or by the second to last addition and the last multiplication, which determine the last addition and therefore k_i, or by the third to last addition and the two last multiplications, and so on. This redundancy enables Eve's model to correctly predict the full secret key even after noise destroys information about multiple operations.

Let us further analyse one of the best models with non-zero average Hamming loss. We present the confusion matrices of the SVM for the scenario with a 8×16 hashing matrix, noise level of 0.1 pA and sampling interval of 20 gates in figure 7. This SVM can predict 7 of the 8 target bits with virtual certainty, while it struggles when predicting the fourth bit of the secret key. By knowing this, when Eve tries to decrypt the messages shared between Alice and Bob, she can use brute force and try all variations of the predicted key with a subset from the power set of the hard-to-predict bits inverted. For instance, in the scenario above using the SVM, she would try the predicted key $k_0k_1k_2k_3k_4k_5k_6k_7$ and then the key $k_0k_1k_2\neg k_3k_4k_5k_6k_7$, where $\neg k_3$ denotes the inversion of k_3. In general, if Eve's model has difficulty predicting k bits of a l-bit secret key, she will often only need to test 2^k key possibilities at most, instead of 2^l. Thus, reasonably-performing models can reduce the search space considerably, making a brute-force approach to secret-key discovery feasible.

4.3. Row models

In the row approach, the best performance was divided between GBM, random forest and decision tree in most scenarios. While the results are generally not as satisfactory as in the full-matrix case, most models were able to predict the full secret key when sampling all gates, and a few still succeeded when sampling 1 of every 3 gates for noise levels up to 0.1 pA.

There was a large discrepancy in performance between the scenarios where we sampled all gates and the ones we did not. Based on this, despite the reduction in irrelevant data fed to the models, the introduction of the neutral data point in the subrows with one less data point, which only happens for sampling intervals larger than 1 gate, worsens the performance of the

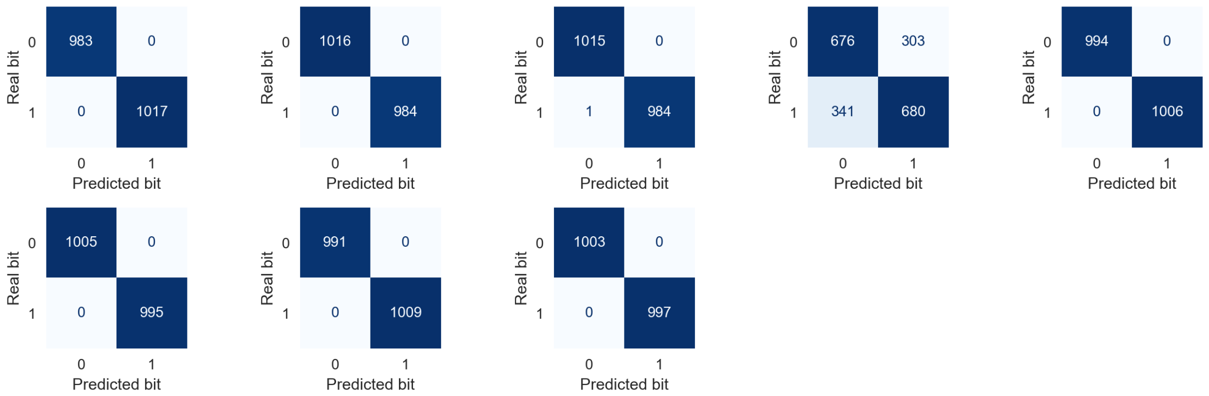
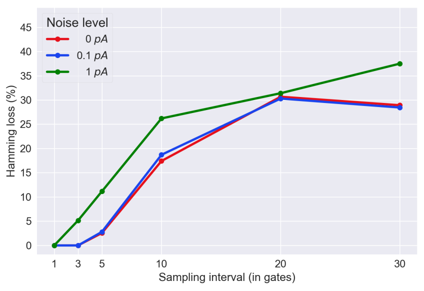
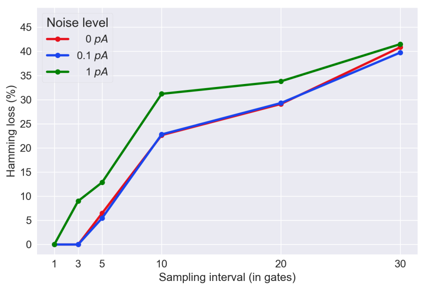


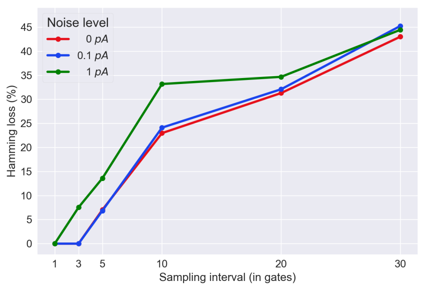
Figure 7: Confusion matrices for a full-matrix SVM trained on the data acquired with a 8×16 hashing matrix, noise level of 0.1 pA and sampling interval of 20 gates.



(a) For a 8×16 hashing matrix.



(b) For a 16×32 hashing matrix.



(c) For a 32×64 hashing matrix.

Figure 8: Average Hamming loss of the predicted keys of the best row model by sampling interval, for each noise level and hashing-matrix size.

models considerably.

In figures 8a to 8c, we present the average Hamming loss of the predicted keys of the best row model for each scenario, plotted by sampling interval. There was no performance degradation with the increase of the hashing-matrix size or noise level when sampling all gates. The same was true for a sampling interval of 3 gates, when the noise level did not go above 0.1 pA. In the remaining scenarios, the degradation with the increase of the hashing-matrix size was not very substantial, while the degradation with the noise level was only significant for high noise levels of 1 pA.

4.4. Attack-approach comparison

The full-matrix approach had generally better results than the row approach. The full-matrix models were able to fully predict the secret key with near certainty for sampling intervals of up to 5 gates while the row models could only do it for sampling intervals up to 3 gates and only if the noise level was low. However, the full-matrix approach is significantly more computationally expensive since the models are trained with much more data. For instance, some of the full-matrix models took a full day to be trained on a computer with a 4-core CPU and 8 Gb of RAM, while the row models took at most around 5 hours. Therefore, the choice of approach usually implies a trade-off between the computational resources needed and the attack performance.

Nevertheless, most of the row models were able to predict the secret key with virtual certainty when sampling all gates. Thus, Eve could use the row approach if she is sure the measuring instrument is has a high-enough sampling rate, since this approach is less time consuming due to the models using much less (irrelevant) features. If she was not able to sample all gates, she would use the full-matrix approach, provided she had enough computational resources. Moreover, if there is a lack of time to perform a similar thorough analysis of a specific physical implementation to know which model to use in a certain scenario, Eve would only use a GBM.

We also tested the full-matrix approach with GBM for a privacy-amplification protocol with a 32×64 randomly-generated binary hashing matrix with no entry restrictions, noise level of 0.1 pA and the same sampling intervals as the previous scenarios, and the results were similar to the 32×64 modified Toeplitz matrix scenarios with the same noise level. Thus, it is highly likely that this attack can be successful for general binary hashing matrices.

4.5. Countermeasures

The results of this analysis make it clear that a similar side-channel attack can be successful. Therefore, it is necessary to consider countermeasures for similar attacks.

Countermeasures can either be applied on the devices of an implementation or on the protocol itself. The first case usually focuses on reducing the side-channel leakage, while the second focuses on eliminating the correlation between the leakage and the secret key.

4.5.1. Noise insertion

A high-enough noise level can always prevent the success of a side-channel attack similar to ours. Thus, a possible countermeasure to such an attack is to attach noise sources to the vulnerable devices of an implementation and make them generate noise in parallel to the main computations. However, it must be ensured that the noise level is high enough. Otherwise, statistical techniques, such as DPA, can be used to isolate the signal from the noise. Moreover, a skilled-enough Eve may be able to circumvent the noise sources if she has access to the implementation. Therefore, proper shielding should also be used to prevent access to the internal circuitry of the implementation devices.

Another option is to modify the protocol to perform randomly-timed dummy operations during the main computations. This acts as noise and successfully hides the main computations from Eve without being easily circumvented. However, it may also lead to a significant reduction in key rate.

4.5.2. Masking and noise insertion

Masking is also known as secret sharing [32] in a more general context. It consists in hiding secret data by not processing it directly. In this context, Alice and Bob split the corrected key into d components or shares such that complete reconstruction of $d - 1$ shares by Eve reveals no information about the secret key. Then, they multiply the hashing matrix with each share separately. The secret key can be reconstructed by logical addition of all the shares due to linearity of the operations.

Masking can be combined with noise to provide information-theoretical security [33]. For a certain amount of noise in the leakages, the secret-key information which Eve can obtain from all leakages of one protocol execution is upper bounded by a value which decreases exponentially with the increase in the number of key shares.

As with simple noise insertion, proper shielding can be crucial for Eve not to circumvent the noise sources.

4.5.3. Randomization

Randomization techniques consist in adding additional steps to the protocol which randomly shuffle the order of operations to eliminate the correlation between side channels and the secret key.

A possible randomization countermeasure is for Alice and Bob to apply independent secret random permutations on the columns of the hashing matrix and on the corrected key before computing their logical multiplication. This effectively renders both attack approaches obsolete. All Eve can do is use brute force and build a different machine-learning model for every possible permutation of either Alice or Bob. For a corrected key with n bits, assuming all columns of the hashing matrix are different, she has to build up to $n!$ models. This is usually a much larger quantity than the number of possibilities for the secret key, 2^l. Thus, this countermeasure recovers the information-theoretical security of the QKD protocol when considering attacks similar to ours. However, it is possible Eve can circumvent it by measuring the power trace of the permutation operations to gain knowledge on the permutation applied.

More aggressive measures, such as Alice and Bob independently applying a different random permutation on the columns of the hashing matrix and on the corrected key for each row of the hashing matrix or applying random permutations on both columns and rows of the hashing matrix, are unnecessary since they would not increase the security further. Moreover, there require them to generate and apply multiple truly-random permutations, which can decrease the key rate unnecessarily.

Yet another countermeasure is to process multiple logical scalar products in parallel to scramble side-channel leakages without necessarily decreasing the key rate.

5. Conclusions

This work demonstrates that machine-learning techniques can be used to robustly characterize the leakages in a QKD implementation and generate powerful attacks.

In it, we analysed a classical-side-channel attack on the privacy-amplification step of a general QKD protocol based on matrix hashing. This attack uses machine-learning techniques to extract information about the secret key from the power-consumption trace of that step. We simulated this attack in multiple scenarios with different sizes of the modified Toeplitz hashing matrix, measurement noise levels and sampling intervals of Eve's measuring instrument, which is a time-independent way of describing the instrument's sampling rate. For all scenarios, we tested two variations of the machine-learning analysis to assess the best approach for Eve.

The overall best-performing approach was training a GBM for each secret-key bit with power traces corresponding to the complete privacy-amplification procedure. The resulting models were able to recover the full secret key for small-enough sampling intervals, equivalent to high-enough sampling rates, with any hashing-matrix size and noise level tested. However, this approach was the most computationally intensive, which leads to a trade-off between the computational resources needed and the attack performance.

In case of non-perfect models, we devised a strategy, based on analysing the confusion matrices of the model, which can make a constrained brute-force search for the secret key feasible.

We also tested our analysis with a general binary hashing matrix and the results were similar to the scenarios with a modified Toeplitz hashing matrix.

Within our simulated scenarios, there was no significant degradation of the performance of the best model with the increase of either the hashing-matrix size or the noise level, provided the sampling interval was small enough. However, a high-enough noise level should always be able to hinder a similar attack. Moreover, the attack performance may degrade for a large-enough hashing matrix. Thus, a more in-depth study of the effects of these two parameters would be useful to understand the limitations of similar attacks.

Due to the success of our attack, we propose multiple countermeasures to similar attacks, based on noise insertion, masking and randomization techniques. Some of them are capable of restoring the information-theoretical security of the QKD protocol, namely masking the key in combination with noise, and applying a secret random permutation on the columns of the hashing matrix and on the corrected key before computing their logical multiplication. Moreover, these countermeasures should also be successful at eliminating other potential side channels in classical-postprocessing devices, such as emanated electromagnetic radiation. Nevertheless, it is crucial to analyse whether these countermeasures are easily circumvented. For instance, the randomization countermeasure might be bypassed by also measuring the power consumption of the permutation operations.

This analysis assumes that Eve has access to the target physical implementation at some point, allowing her to characterize its internal structure and replicate it. We should not discard the possibility of such scenario. In fact, this is specially feasible in QKD networks with remotely-located repeaters, where constant surveillance can be difficult.

Currently, a quantum cryptosystem whose security relies solely on quantum postulates is unrealistic. Most physical implementations are imperfect and naturally present both classical and quantum side channels. Our work reinforces that classical side channels in implementations must be taken into account for practical use of QKD. We note that this argument remains relevant for MDI- and DI-QKD implementations.

Nevertheless, we emphasize that QKD still has a clear advantage over classical systems based on computational security, since the security of the former only depends on Eve successfully performing a side-channel attack, while the latter can be broken either by side-channel attacks or by Eve saving the encrypted messages until she has enough computational resources to decrypt them.

The attack in this work was performed on simulated power traces due to lack of access to physical hardware. A natural next step is to use the same machine-learning framework to analyse power traces from a physical implementation to verify the validity of our analysis. We believe this can only strengthen our argument for considering classical side channels in QKD implementations.

Besides power consumption, our attack framework can be used to exploit other side channels, such as electromagnetic radiation emanated from implementation devices and even their cache state throughout the QKD protocol. All potential side channels should be thoroughly explored.

This work also led to the paper [34], currently awaiting publishing.

References

[1] S. Pirandola et al. *Advances in Optics and Photonics* 12.4 (2020), p. 1012. DOI: 10.1364/aop.361502.

[2] L. Salvail et al. 2009. arXiv: 0904.4072 [quant-ph].

[3] H.-J. Briegel et al. *Physical Review Letters* 81.26 (1998), pp. 5932–5935. DOI: 10.1103/PhysRevLett.81.5932.

[4] H.-K. Lo et al. *Physical Review Letters* 108.13 (2012), p. 130503. DOI: 10.1103/PhysRevLett.108.130503.

[5] A. Acín et al. *Physical Review Letters* 98.23 (2007), p. 230501. DOI: 10.1103/PhysRevLett.98.230501.

[6] A. Lamas-Linares and C. Kurtsiefer. *Optics Express* 15.15 (2007), p. 9388. DOI: 10.1364/oe.15.009388.

[7] D. Park et al. *ICT Express* 7.1 (2021), pp. 36–40. DOI: 10.1016/j.icte.2021.01.013.

[8] G. Kim et al. *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. 2021, pp. 257–261. DOI: 10.1109/ICTC52510.2021.9620820.

[9] S. Kim et al. *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. 2018, pp. 736–739. DOI: 10.1109/ICTC.2018.8539703.

[10] K. Durak and N. Jam. 2020. arXiv: 2004.14445 [quant-ph].

[11] A. Weber et al. *Computer Security – ESORICS 2021*. 2021, pp. 235–256. DOI: 10.1007/978-3-030-88428-4_12.

[12] J. F. Bravo. Master's thesis. 2022. URL: https://fenix.tecnico.ulisboa.pt/cursos/meft/dissertacao/1691203502344507.

[13] J. Carter and M. N. Wegman. *Journal of Computer and System Sciences* 18.2 (1979), pp. 143–154. DOI: 10.1016/0022-0000(79)90044-8.

[14] M. Tomamichel and A. Leverrier. *Quantum* 1 (2017), p. 14. DOI: 10.22331/q-2017-07-14-14.

[15] C. H. Bennett et al. *SIAM Journal on Computing* 17.2 (1988), pp. 210–229. DOI: 10.1137/0217014.

[16] F. Xu et al. *Reviews of Modern Physics* 92.2 (2020), p. 025002. DOI: 10.1103/RevModPhys.92.025002.

[17] H. Krawczyk. *Advances in Cryptology — CRYPTO '94*. 1994, pp. 129–139. DOI: 10.1007/3-540-48658-5_15.

[18] M. Hayashi. *IEEE Transactions on Information Theory* 57.6 (2011), pp. 3989–4001. DOI: 10.1109/TIT.2011.2110950.

[19] F.-X. Standaert et al. *Advances in Cryptology - EUROCRYPT 2009*. 2009, pp. 443–461. DOI: 10.1007/978-3-642-01001-9_26.

[20] K. P. Murphy. *Machine learning: A Probabilistic Perspective*. 2012.

[21] E. Fix and J. L. Hodges. *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), p. 238. DOI: 10.2307/1403797.

[22] B. E. Boser et al. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. 1992, pp. 144–152. DOI: 10.1145/130385.130401.

[23] L. Breiman. *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[24] J. H. Friedman. *Computational Statistics & Data Analysis* 38.4 (2002), pp. 367–378. DOI: 10.1016/S0167-9473(01)00065-2.

[25] C. M. Bishop. *Pattern Recognition and Machine Learning*. 1st. 2006.

[26] M. Feurer and F. Hutter. *Automated Machine Learning: Methods, Systems, Challenges*. 2019, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1.

[27] L. Li et al. *Proceedings of Machine Learning and Systems*. Vol. 2. 2020, pp. 230–246. URL: https://proceedings.mlsys.org/paper/2020/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html.

[28] F. Pedregosa et al. *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[29] P. Kocher et al. *Advances in Cryptology — CRYPTO' 99*. 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.

[30] A. Wiltgen et al. *26th Symposium on Integrated Circuits and Systems Design (SBCCI)*. 2013, pp. 1–6. DOI: 10.1109/SBCCI.2013.6644863.

[31] C. D. Motchenbacher and J. A. Connelly. *Low-Noise Electronic System Design*. 1st. 1993.

[32] A. Shamir. *Communications of the ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.

[33] E. Prouff and M. Rivain. *Advances in Cryptology – EUROCRYPT 2013*. 2013, pp. 142–159. DOI: 10.1007/978-3-642-38348-9_9.

[34] J. F. Bravo and P. Mateus. *Entropy* 24.12 (2022). (to be published).