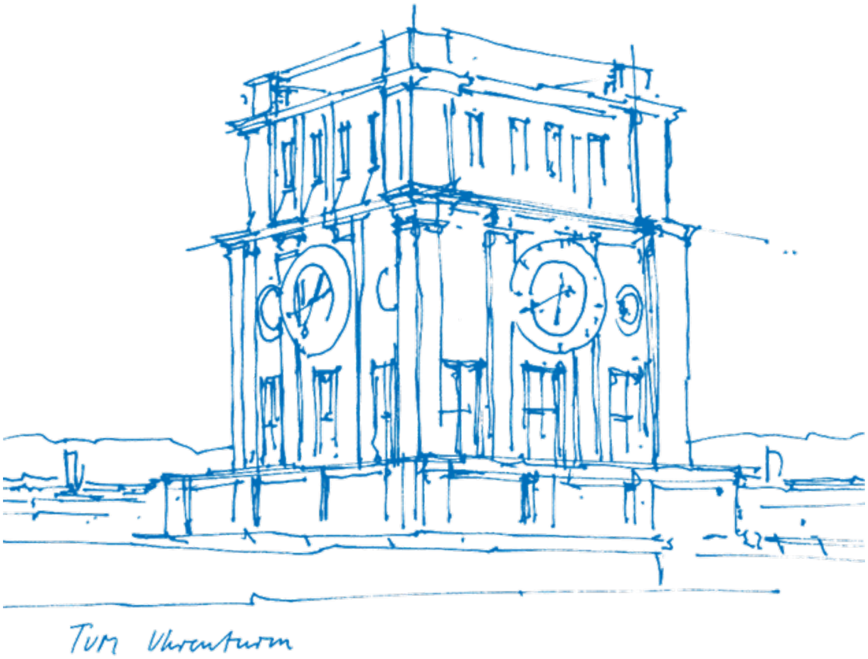
Chair for Security in Information Technology
School of Computation, Information and Technology
Technical University of Munich

Towards Efficient Helper Data Algorithms for Multi-Bit PUF Quantization

Marius Drechsler



Chair for Security in Information Technology
School of Computation, Information and Technology
Technical University of Munich

Towards Efficient Helper Data Algorithms for Multi-Bit PUF Quantization

Marius Drechsler

Thesis for the attainment of the academic degree

Bachelor of Science (B.Sc.)

at the School of Computation, Information and Technology of the Technical University of Munich.

Examiner:
Prof. Dr. Georg Sigl

Supervisor:
M.Sc. Jonas Ruchti

Submitted:
Munich, 02.09.2024

Contents

1 Introduction

In the field of cryptography, physical unclonable function (PUF) devices are a popular tool for key generation and storage [1], [2]. In general, a PUF refers to a type of circuit that exhibits slightly different behaviors during operation due to minor variations in the manufacturing process. Since the behaviour of one PUF device is now only reproducible on itself and not on a device of the same type with the same manufacturing process, it can be used for secure key generation and/or storage.

To improve the reliability of the keys generated and stored using the PUF, various helper data algorithms (HDAs) have been introduced. The general operation of a PUF with a HDA can be divided into two separate stages: *enrollment* and *reconstruction* as shown in Figure 1 [3].

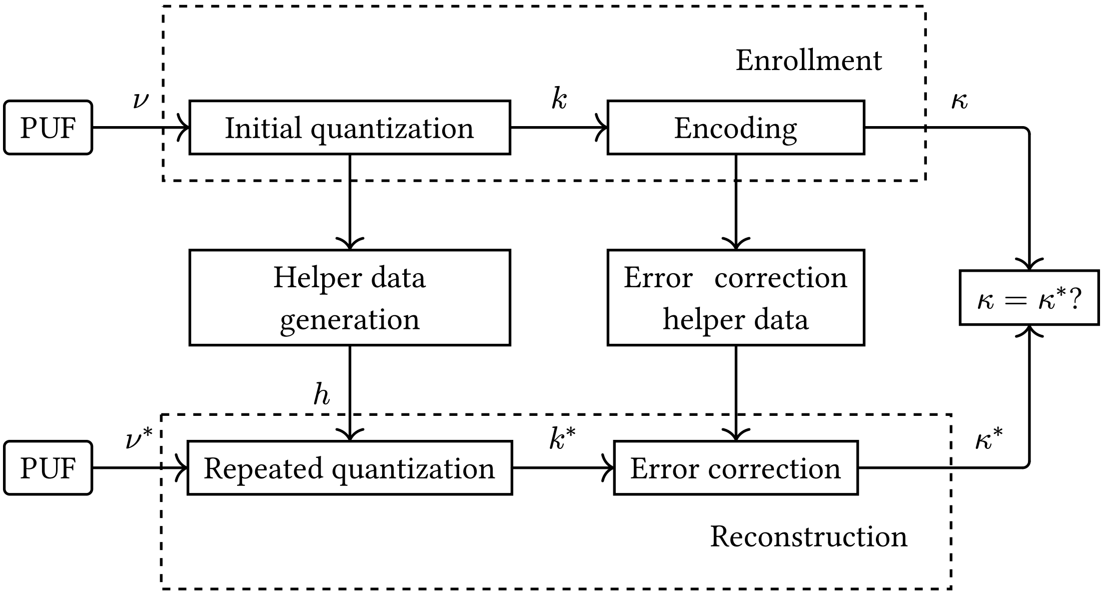


Figure 1: PUF model description using enrollment and reconstruction.

The enrollment stage will usually be performed in near ideal, lab-like conditions i.e. at room temperature ($25°C$). During this phase, a first PUF readout ν with corresponding helper data h is generated. Going on, reconstruction can now be performed under varying conditions, for example at a higher temperature. Here, slightly different PUF readout ν^* is generated. Using the helper data h the new PUF readout ν^* can be improved to be less deviated from v as before. One possible implementation of this principle is called *Fuzzy Commitment* [4], [5].

Previous works already introduced different HDAs with various strategies [6], [7]. The simplest form of helper-data one could generate is reliability information for every PUF bit. Here, the HDA marks unreliable PUF bits that are then either discarded during reconstruction or rather corrected using an error correction code after the quantization process.

Going on, publications [8] and [9] introduced a metric-based HDA as Two Metric Helper Data method (TMHD). The main goal of such a HDA is to improve the reliability of the PUF during the quantization step of the enrollment phase. To achieve that, helper data is generated to define multiple quantizers for the reconstruction phase to minimize the risk of bit errors. A generalization outline to extend TMHD for higher order bit quantization has already been proposed by Fischer in [10].

In the course of this work, we will first take a closer look at SMHD as proposed by Fischer [10] and provide a concrete realization for this method. We will also propose a method to shape the input values of a PUF to better fit within the bounds of a multi-bit quantizer which we call Boundary Adaptive Clustering with Helper data (BACH). We will investigate the question which of these two HDAs provides the better performance for higher order bit cases with the least amount of helper data bits.

1.1 Notation

To ensure a consistent notation of functions and ideas, we will now introduce some conventions and definitions.

Random distributed variables will be notated with a capital letter, i.e. X. Realizations will be the corresponding lower case letter, x. Values of x subject to some kind of error are marked with a $*$ in the exponent e.g., x^*. Vectors will be written in bold text: e.g., $\boldsymbol{k}$ represents a vector of quantized symbols. Matrices are denoted with a bold capital letter: $\boldsymbol{M}$. We will call a quantized symbol k. k consists of all possible binary symbols, i.e. $0, 01, 110$. A quantizer will be defined as a function $\mathcal{Q}(x, \boldsymbol{a})$ that returns a quantized symbol k. We also define the following special quantizers for metric based HDAs: A quantizer used during the enrollment phase is defined by a calligraphic $\mathcal{E}$. For the reconstruction phase, a quantizer will be defined by a calligraphic $\mathcal{R}$ Figure 2 shows the curve of a 2-bit quantizer that receives $\tilde{x}$ as input. In the case, that the value of $\tilde{x}$ equals one of the four bounds, the quantized value is chosen randomly from the relevant bins.

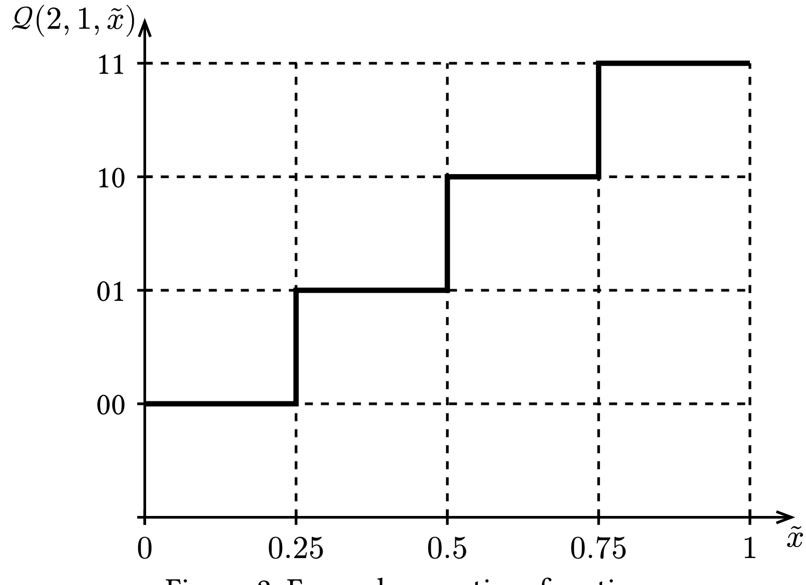


Figure 2: Example quantizer function

For the S-Metric Helper Data Method, we introduce a function

$$\mathcal{Q}(S, M), \tag{1}$$

where S determines the number of metrics and M the bit width of the symbols. The corresponding metric is defined through the lower case s, the bit symbol through the lower case m. To compare both SMHD and BACH, we will use a ratio $\mathscr{r} = \frac{\text{Extracted bits}}{\text{Helper data bits}}$. This ratio gives us an idea how many helper data bits were used to obtain a quantized symbol. $\mathscr{r}$ is smaller than 1 if the amount of helper data bits per quantized symbol is bigger than the symbol bit width itself and bigger than 1 otherwise.

1.1.1 Tilde Domain

The tilde domain describes the range of numbers between 0 and 1, which is defined by the image of a cumulative distribution function (CDF). As also described in [10], we will use a CDF to transform the real PUF values into the tilde domain. This transformation can be performed using the function $\xi = \tilde{x}$. The key property of this transformation is the resulting uniform distribution of x.

Considering a normal distribution, the CDF is defined as

$$\xi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{2}$$

Empirical cumulative distribution function (eCDF)

We will not always be able to find an analytical description of a probability distribution and its corresponding CDF. Alternatively, an eCDF can be constructed through sorting the empirical measurements of a distribution [11]. Although less accurate, this method allows a more simple and less computationally complex way to transform real valued measurements into the tilde domain. We will mainly use the eCDF in Section 2 because of the difficulty of finding an analytical description for the CDF of a weighted linear combination of random variables. The function for an eCDF can be defined as

$$\xi_{\text{eCDF}}(x) = \frac{\text{number of elements in } \boldsymbol{z}, \text{ s.t} \leq x}{n} \in [0, 1], \tag{3}$$

where n defines the number of elements in the vector $\boldsymbol{z}$. If the vector $\boldsymbol{z}$ were to contain the elements $[1, 3, 4, 5, 7, 9, 10]$ and $x = 5$, Equation 3 would result to $\xi_{\text{eCDF}}(5) = \frac{4}{7}$.
The application of Equation 3 on X will transform its values into the empirical tilde domain.

We can also define an inverse eCDF:

$$\xi_{\text{eCDF}}^{-1}(\tilde{x}) = \tilde{x} \cdot n \tag{4}$$

The result of Equation 4 is the index i of the element z_i from the vector of realizations $\boldsymbol{z}$.

To apply the eCDF to our numerical results later, we will sort the vector of realizations $\boldsymbol{z}$ of a random distributed variable Z in ascending order.

2 S-Metric Helper Data Method

A metric based HDA generates helper data at PUF enrollment to provide more reliable results at the reconstruction stage. Each of these metrics correspond to a quantizer with different bounds to lower the risk of bit or symbol errors during reconstruction. For this kind of HDA, the generated metric is used as helper data and thus does not have to be kept secret.

2.1 Background

Before we turn to a concrete realization of the S-Metric method, let's take a look at its predecessor, the Two-Metric Helper Data Method.

2.1.1 Two-Metric Helper Data Method

The simplest form of a metric-based HDA is the Two-Metric Helper Data Method. Its quantization only yields symbols of 1-bit width and it only uses a single bit of helper data to store the choice of metric.

Figure 4 and Figure 5 illustrate an example enrollment and reconstruction process. Consider the marked point the value of the initial measurement and the marked range our margin of error. If we now were to use the original quantizer shown in Figure 4 during both the enrollment and the reconstruction phases, we would risk a bit error, because the margin of error overlaps with the lower quantization bound $-a$, which we can call a point of uncertainty. To alleviate this we generated helper data during enrollment as depicted in Figure 6, we can make use of a different quantizer $\mathcal{R}(1, 2, x)$ whose boundaries do not overlap with the error margin.

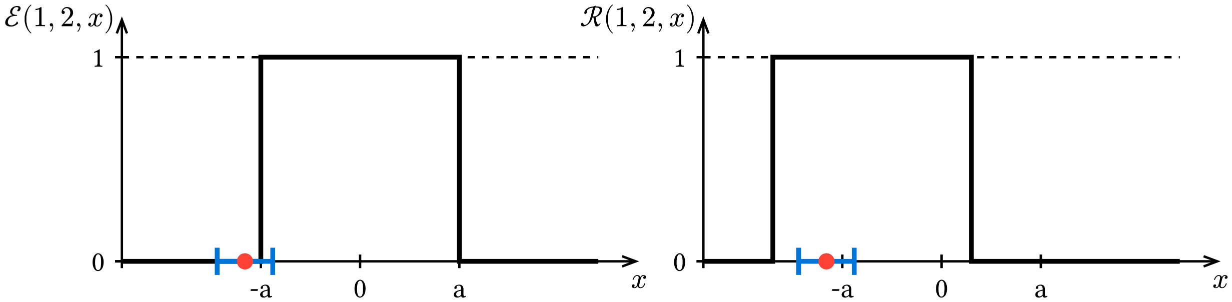


Figure 4: Example enrollment

Figure 5: Example reconstruction

Figure 3: Example enrollment and reconstruction of TMHD. The window function describes the quantizer used to define the resulting bit. The red dot shows a possible PUF readout measurement with its blue marked strip as margin of error.

Publications [8] and [9] find all the relevant bounds for the enrollment and reconstruction phases under the assumption that the PUF readout (our input value x) is zero-mean Gaussian distributed. Because the parameters for symbol width and number of metrics always stay the same, we can – without loss of generality – assume the standard deviation as $\sigma = 1$ and calculate the bounds for 8 equi-probable areas for this distribution. This is done by finding two bounds a and b such, that

$$\int_a^b f_{X(x)} dx = \frac{1}{8} \tag{5}$$

This operation yields 9 bounds defining these areas $-\infty$, $-T1$, $-a$, $-T2$, 0, $T2$, a, $T1$ and $+\infty$. During the enrollment phase, we will use $\pm a$ as our quantizing bounds, returning 0 if the absolute value of x is smaller than a and 1 otherwise. The corresponding metric is chosen based on the following conditions:

$$M = \begin{cases} M1, x < -a \lor 0 < x < a \\ M2, -a < x \lor 1 < a < x \end{cases}. \tag{6}$$

Figure 6 shows the curve of a quantizer $\mathcal{Q}$ that would be used during the Two-Metric enrollment phase.

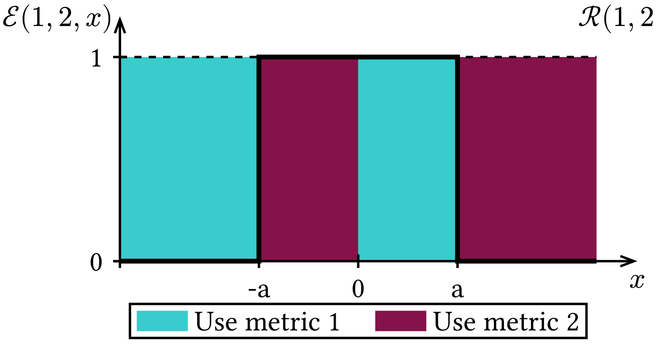


Figure 6: Two-Metric enrollment

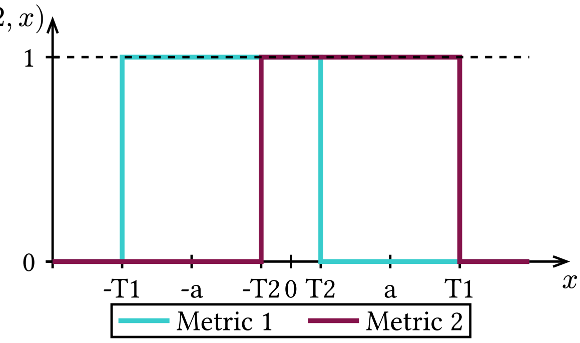


Figure 7: Two-Metric reconstruction

As previously described, each of these metrics correspond to a different quantizer. In the reconstruction phase, we can use the generated helper data and define a reconstructed bit based on the chosen metric as follows:

$$M1 : k = \begin{cases} 0, x < T1 \lor T2 < x \\ 1, -T1 < x < T2 \end{cases}, \qquad M2 : k = \begin{cases} 0, x < -T2 \lor T1 < x \\ 1, -T2 < x < T1 \end{cases}.$$

Figure 7 illustrates the basic idea behind the Two-Metric method. Using the helper data, we will move the bounds of the original quantizer (Figure 4) one octile to each side, yielding two new quantizers. The advantage of this method comes from moving the point of uncertainty away from our enrollment-time readout.

2.1.2 S-Metric Helper Data method (SMHD)

Going on, the Two-Metric Helper Data Method can be generalized as shown in [10]. This generalization allows for higher-order bit quantization and the use of more than two metrics.

A key difference to the Two-Metric approach is the alignment of quantization areas. Methods described in [8] and [9] use two bounds for 1-bit quantization, namely $\pm a$. Contrary, the method introduced by Fischer in [10] would look more like a sign-based quantizer if the configuration $\mathcal{Q}(2, 1)$ is used, using only one quantization bound at $x = 0$. Figure 8 and Figure 9 illustrate this difference, .

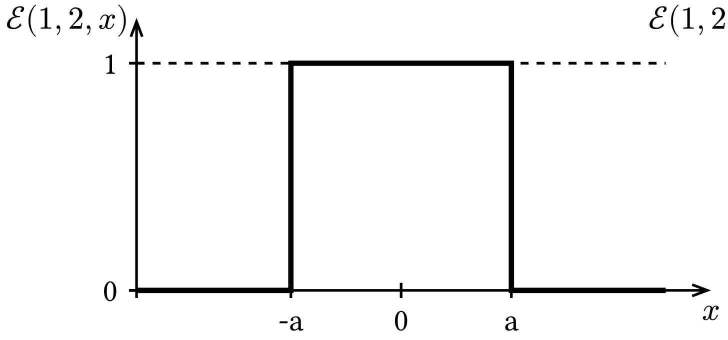


Figure 8: Two-Metric enrollment

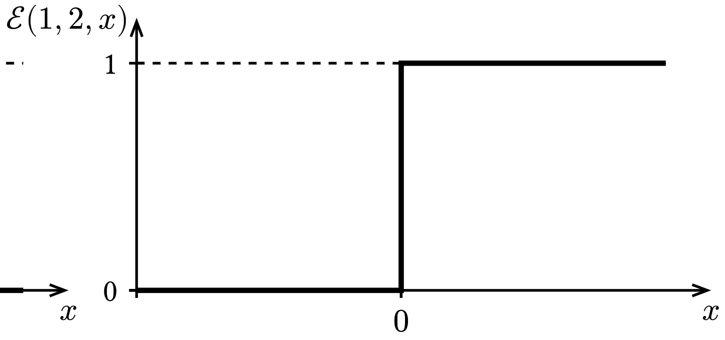


Figure 9: S-Metric enrollment with 1-bit configuration

The generalization consists of two components:

- **Higher-order bit quantization**
  We can introduce more steps to our quantizer and use them to extract more than one bit out of our PUF readout.
- **More than two metrics**
  Instead of splitting each quantizer into only two equi-probable parts, we can increase the number of metrics at the cost of generating more helper data to increase reliability.

2.2 Realization

We will now propose a specific realization of the S-Metric Helper Data Method.
Instead of using the PUF readout directly for SMHD, we can use a CDF to transform these values into the tilde domain. The only requirement we would need to meet here is that the CDF of the probability distribution used is known. This allows us to use equi-distant bounds for the quantizer instead of equi-probable ones.

From now on we will use the following syntax for quantizers that use the S-Metric Helper Data Method:

$$\mathcal{Q}(S, M, \tilde{x}), \tag{8}$$

where S defines the number of metrics, M the number of bits and $\tilde{x}$ a Tilde-Domain transformed PUF measurement.

2.2.1 Enrollment

To enroll our PUF key, we will first need to define the quantizer for higher order bit quantization and helper data generation. Because our transformed PUF readout $\tilde{x}$ can be interpreted as a realization of a uniformly distributed variable $\tilde{X}$, we can define the width Δ of our quantizer bins as follows:

$$\Delta = \frac{1}{2^M}. \tag{9}$$

For example, if we were to extract a symbol with the width of 2 bits from our PUF readout, we would need to evenly space $2^2 = 4$ bins. Using equation Equation 9, the step size for a 2-bit quantizer would result to:

$$\Delta' = \left.\frac{1}{2^M}\right|_{M=2} = \frac{1}{4}. \tag{10}$$

Figure 10 shows a plot of the resulting quantizer function that would yield symbols with two bits for one measurement $\tilde{x}$.

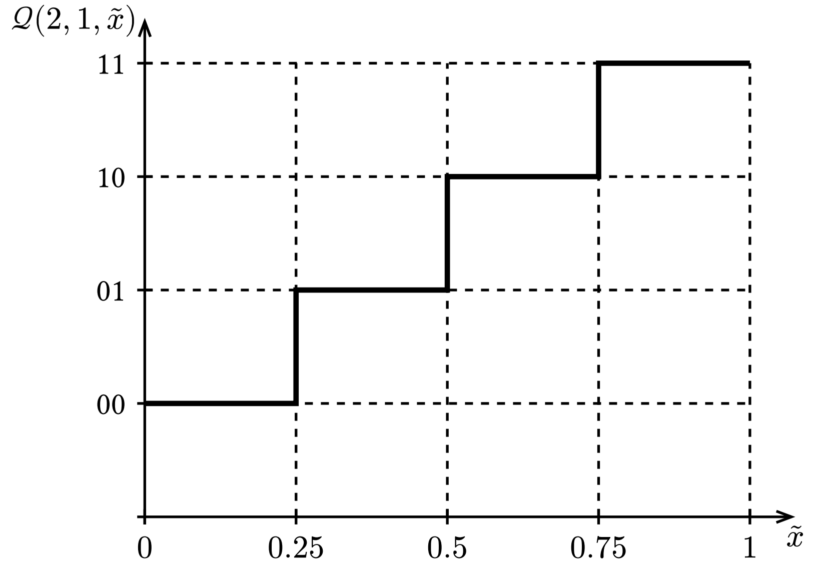


Figure 10: 2-bit quantizer

Right now, this quantizer wouldn't help us generating any helper data. To achieve that, we will need to divide a symbol step – one, that returns the corresponding quantized symbol - into multiple substeps. Using S, we can define the step size Δ_S as the division of Δ by S:

$$\Delta_S = \frac{\Delta}{S} = \frac{1}{2^M \cdot S} \tag{11}$$

We can now redefine our previously defined quantizer function to not only return the quantized symbol, but a tuple consisting of the quantized symbol and the metric ascertained that we will save as helper data for later.

Going on in our example, we could choose the amount of our metrics to be 2. According to Equation 11, we would then half our step size:

$$\Delta'_S = \left.\frac{\Delta'}{S}\right|_{S=2} = \frac{1}{4 \cdot 2} = \frac{1}{8} \tag{12}$$

This means, we can update our quantizer function with the new step size $\Delta'_S = \frac{1}{8}$ and redefining its output as a tuple consisting of bit value and helper data.

We can visualize the quantizer that we will use during the enrollment phase of a 2-bit 2-metric configuration as depicted in Figure 11.

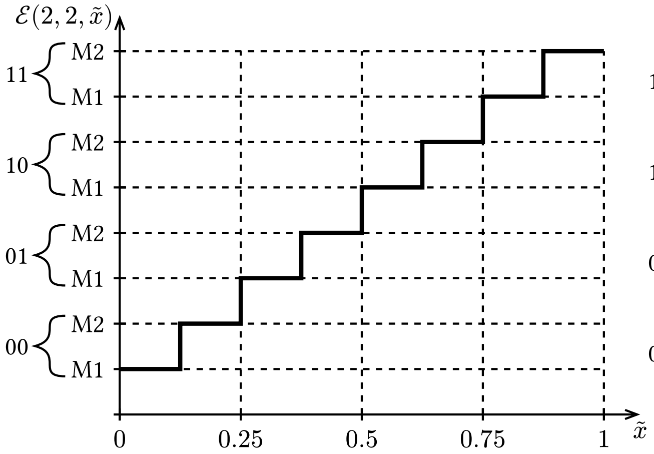


Figure 11: 2-bit 2-metric enrollment

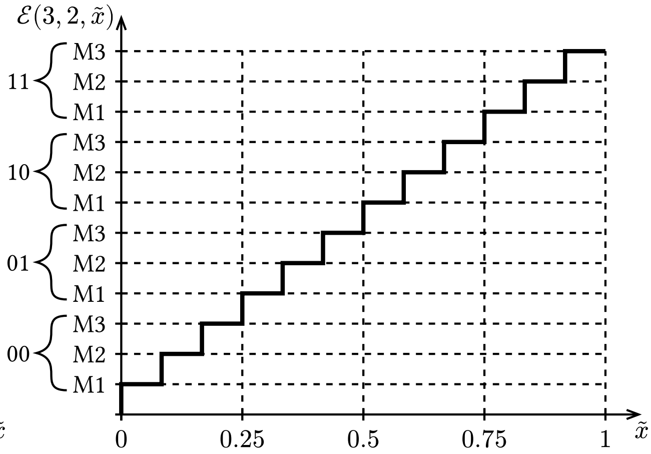


Figure 12: 2-bit 3-metric enrollment

To better demonstrate the generalization to S-metrics, Figure 12 shows a 2-bit quantizer that generates helper data based on three metrics instead of two. In that sense, increasing the number of metrics will increase the number of sub-steps for each symbol.

We can now perform the enrollment of a full PUF readout. Each measurement will be quantized with out quantizer $\mathcal{E}$, returning a tuple consisting of the quantized symbol and helper data.

$$\kappa_i = \mathcal{E}(s, m, \tilde{x_i}) = (k, h)_i \quad . \tag{13}$$

Performing the operation of Equation 13 for our whole set of measurements will yield a vector of tuples $\boldsymbol{\kappa}$.

2.2.2 Reconstruction

We already demonstrated the basic principle of the reconstruction phase in section Section 2.1.1, which showed the advantage of using more than one quantizer during reconstruction.

We will call our repeated measurement of $\tilde{x}$ that is subject to a certain error $\tilde{x^*}$. To perform reconstruction with $\tilde{x^*}$, we will first need to find all S quantizers for which we generated the helper data in the previous step and then choose the one corresponding to the saved metric.

We have to distinguish the two cases, that S is either even or odd:
If S is even, we need to define S quantizers offset by multiples of φ. We can define the ideal position for the quantizer bounds based on its corresponding metric as centered around the center of the metric.

We can find these new bounds graphically as depicted in Figure 13. We first determine the x-values of the centers of a metric (here M1, as shown with the arrows). We can then place the quantizer steps with step size Δ (Equation 9) evenly spaced around these points. If the resulting quantizer bound is smaller than 0 or bigger than 1, we will either add or subtract 1 from its value so it stays in the defined range of the tilde domain. With these new points for the vertical steps of $\mathcal{Q}$, we can draw the new quantizer for the first metric in Figure 14.

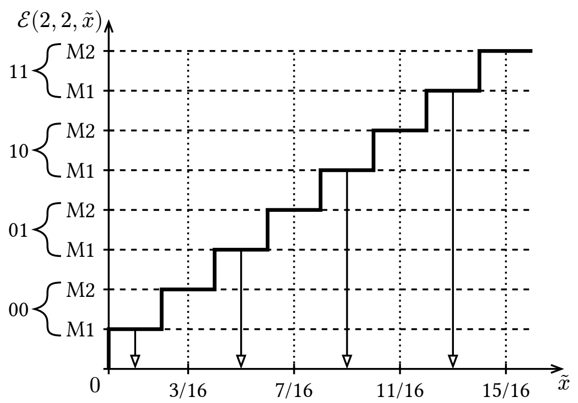


Figure 13: Ideal centers and bounds for the M1 quantizer

$\Rightarrow$

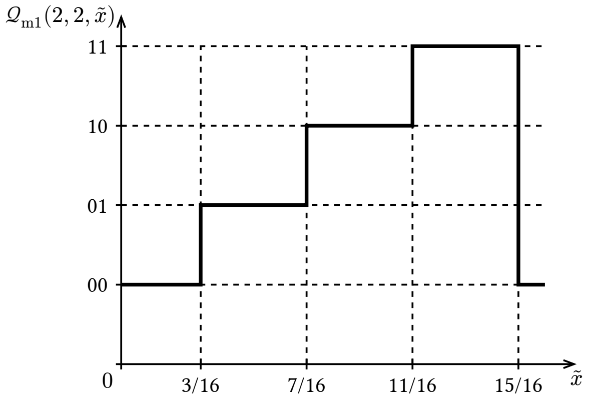


Figure 14: Quantizer for the first metric

As for metric 2, we can apply the same strategy and find the points for the vertical steps to be at $\frac{1}{16}$, $\frac{5}{16}$, $\frac{9}{16}$ and $\frac{13}{16}$. This quantizer is shown together with the first-metric quantizer in Figure 15, forming the complete quantizer for the reconstruction phase of a 2-bit 2-metric configuration $\mathcal{R}(2,2,\tilde{x})$.

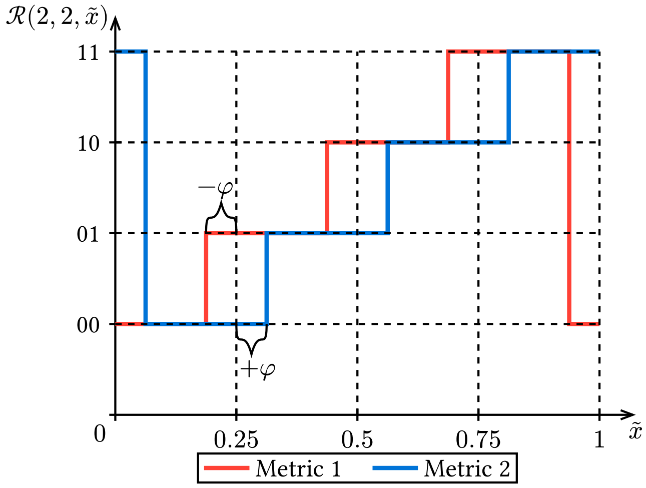


Figure 15: 2-bit 2-metric reconstruction quantizer

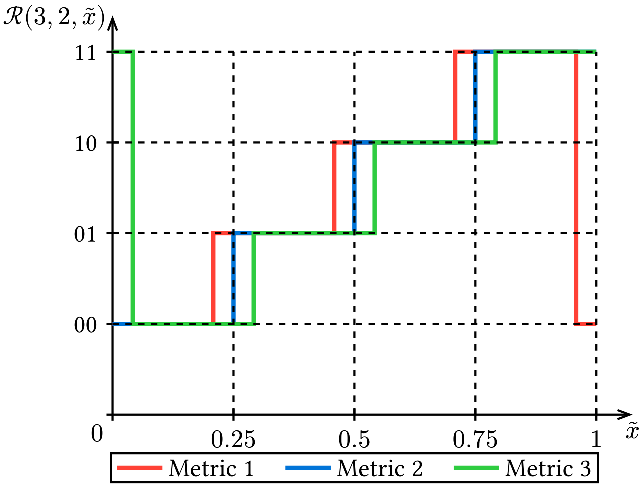


Figure 16: 2-bit 3-metric reconstruction quantizer

Analytically, the offset we are applying to $\mathcal{E}(2,2,\tilde{x})$ can be defined as

$$\Phi = \left.\frac{1}{2^M \cdot S \cdot 2}\right|_{M=2,S=2} = \frac{1}{16} \;. \tag{14}$$

Φ is the constant that we will multiply with a certain metric index $i \in \left[-\frac{S}{2}, ..., \frac{S}{2}\right]$ to obtain the metric offset φ, which is used to define each of the S different quantizers for reconstruction. In Figure 15, the two metric indices $i = \pm 1$ will be multiplied with Φ, yielding two quantizers, one moved $\frac{1}{16}$ to the left and one moved $\frac{1}{16}$ to the right.

If a odd number of metrics is given, the offset can still be calculated using Equation 14. Additionally, we will keep the original quantizer used during enrollment as the quantizer for metric $\frac{s-1}{2}$ (Figure 16).

To find all metric offsets for values of $S > 3$, we can use Algorithm 1. We can calculate φ based on S and M using Equation 14. The resulting list of offsets is correctly ordered and can be mapped to the corresponding metrics in ascending order.

Algorithm 1: Find all offsets φ

```
input $\Phi, S$
list offsets $\varphi$
if $S$ is odd
    $S = s - 1$
    append 0 to list offsets
while $i \leq \frac{S}{2}$
    append $+(i \cdot \Phi)$ to list offsets
    append $-(i \cdot \Phi)$ to list offsets
sort list offsets in ascending order
return offsets
end
```

Offset properties

Before we go on and experimentally test this realization of the S-Metric method, let's look deeper into the properties of the metric offset value φ. Comparing Figure 15, Figure 16 and their respective values of Equation 14, we can observe, that the offset Φ gets smaller the more metrics we use.

Table 1: Offset values for 2-bit configurations

M	1	2	3	4	5	6	7	8	9	10
Φ	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{24}$	$\frac{1}{32}$	$\frac{1}{40}$	$\frac{1}{48}$	$\frac{1}{56}$	$\frac{1}{64}$	$\frac{1}{72}$	$\frac{1}{80}$

As previously stated, we will need to define S quantizers, $\frac{S}{2}$ times to the left and $\frac{S}{2}$ times to the right. For example, setting the parameter S to 4 means we will need to move the enrollment quantizer 2 times to the left and right. As we can see in Table 2, φ for the maximum metric indices $i = \pm 2$ are identical to the offsets of a 2-bit 2-metric configuration. In fact, this property carries on for higher even numbers of metrics, as shown in Table 3.

Table 2: 2-bit 4-metric offsets

i	-2	-1	1	2
Metric	M1	M2	M3	M4
φ	$-\frac{1}{16}$	$-\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{16}$

Table 3: 2-bit 6-metric offsets

i	-3	-2	-1	1	2	3
Metric	M1	M2	M3	M4	M5	M6
φ	$-\frac{1}{16}$	$-\frac{1}{24}$	$-\frac{1}{48}$	$\frac{1}{48}$	$\frac{1}{24}$	$\frac{1}{16}$

At $s = 6$ metrics, the biggest metric offset we encounter is $\varphi = \frac{1}{16}$ at $i = \pm 3$.
This biggest (or maximum) offset is of particular interest to us, as it tells us how far we deviate from the original quantizer used during enrollment. The maximum offset for a 2-bit configuration φ is $\frac{1}{16}$ and we only introduce smaller offsets in between if we use a higher even number of metrics.

More formally, we can define the maximum metric offset as follows:

$$\varphi_{\max} = \frac{\left\lfloor \frac{S}{2} \right\rfloor}{2^M \cdot S \cdot 2} \tag{15}$$

It is important to note, that $\varphi_{\max}$ is dependent on the parameter S if S is an odd number.

Table 4: 2-bit maximum offsets, odd

S	3	5	7	9
$\boldsymbol{\varphi}_{\mathbf{max,odd}}$	$\frac{1}{24}$	$\frac{1}{20}$	$\frac{3}{56}$	$\frac{1}{18}$

The higher S is chosen, the closer we approximate $\varphi_{\max}$ for even choices of S, as shown in Equation 16.1. This means, while also keeping the original quantizer during the reconstruction phase, the maximum offset for an odd number of metrics will always be smaller than for an even number.

$$\lim_{S \to \infty} \varphi_{\text{max,odd}} = \frac{\left\lfloor \frac{S}{2} \right\rfloor}{2^M \cdot S \cdot 2} = \frac{S-1}{2^M \cdot S \cdot 4} \tag{16.1}$$

$$= \frac{1}{2^M \cdot 4} = \varphi_{\text{max,even}} \tag{16.2}$$

Because $\varphi_{\text{max,odd}}$ only approximates $\varphi_{\text{max,even}}$ if $S \to \infty$ we can assume, that configurations with an even number of metrics will always perform marginally better than configurations with odd numbers of metrics because the bigger maximum offset allows for better reconstructing capabilities.

2.3 Improvements

The S-Metric Helper Data Method proposed by Fischer in [10] can be improved by using Gray-coded labels for the quantized symbols instead of naive labelling.

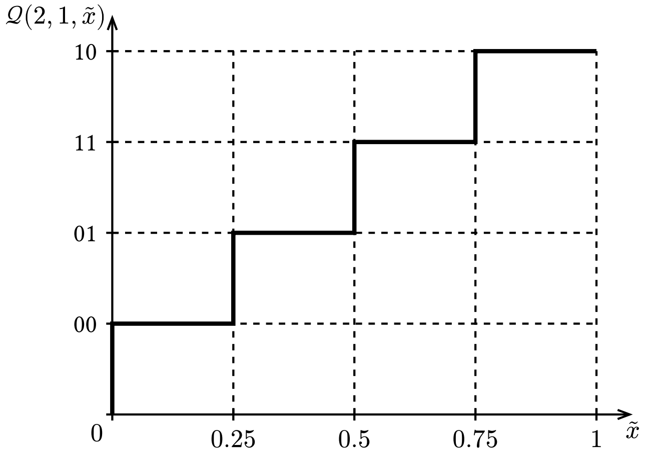


Figure 17: Gray Coded 2-bit quantizer

Figure 17 shows a 2-bit quantizer with gray-coded labelling. In this example, we have an advantage at $\tilde{x} \approx 0.5$, because a quantization error only returns one wrong bit instead of two.

Furthermore, the transformation into the Tilde-Domain could also be performed using the eCDF to achieve a more precise uniform distribution because we do not have to estimate a standard deviation of the input values.

2.4 Experiments

We tested the implementation of Section 2.2 with the dataset of [12]. The dataset contains counts of positives edges of a ring oscillator at a set evaluation time D. Based on the count and the evaluation time, the frequency of a ring oscillator can be calculated using: $f = 2 \cdot \frac{k}{D}$. Because we want to analyze the performance of the S-Metric method over different temperatures, both during enrollment and reconstruction, we are limited to the experimental measurements of [12] which varied the temperature during the FPGA operation. We will have measurements of 50 FPGA boards available with 1600 and 1696 ring oscillators each. The two measurement sets are obtained from different slices of the FPGA board where the only difference to note is the number of ring oscillators available. To obtain the values to be processed, we subtract them in pairs, yielding 800 and 848 ring oscillator frequency differences *df*.
Because we can assume that the frequencies *f* are i.i.d., the difference *df* can also be assumed to be i.i.d. To apply the values *df* to our implementation of the S-Metric method, we will first transform them into the Tilde-Domain using an inverse CDF, resulting in uniform distributed values $\tilde{x}$. Our resulting dataset consists of bit error rates (BERs) for quantization symbol widths of up to 6 bits evaluated with generated helper-data from up to 100 metrics. In the following section, we will often set the maximum number of metrics to be $S = 100$. This choice refers to the asymptotic behaviour of the BER and can be equated with the choice $S \to \infty$.

2.4.1 Results & Discussion

The bit error rate of different S-Metric configurations for naive labelling can be seen in Figure 18. For this analysis, enrollment and reconstruction were both performed at room temperature.

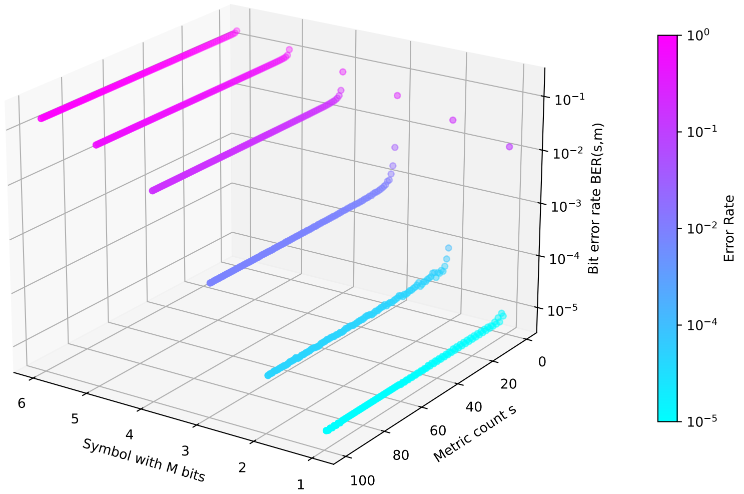


Figure 18: Bit error rates for same-temperature execution. Here we can already observe the asymptotic BERs for higher metric numbers. The error rate is scaled logarithmically here.

We can observe two key properties of the S-Metric method in Figure 18. The exponential growth of the BER can be observed if we set $S = 1$ and increase M up to 6. Also, as we expanded on in Section 2.2.2.1, at some point using more metrics will no longer improve the bit error rate of the key. At a symbol width of $M \geq 6$ bits, no further improvement through the S-Metric method can be observed.

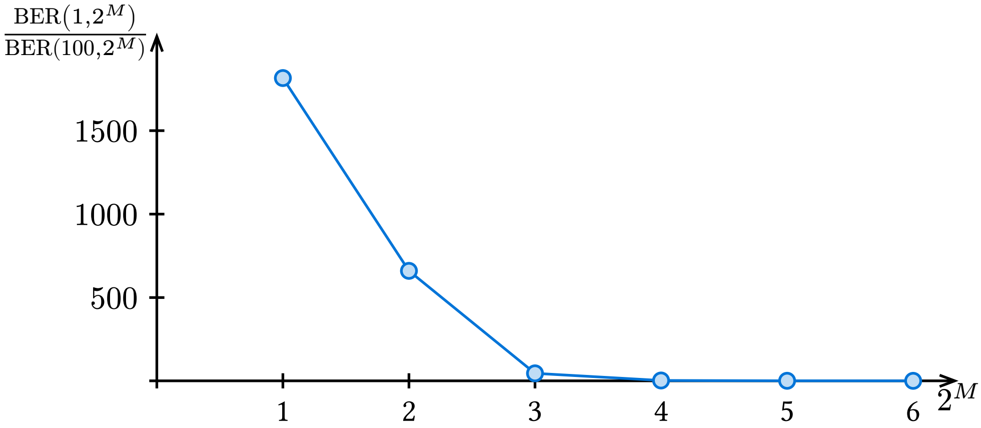


Figure 19: Asymptotic performance of SMHD

This tendency can also be shown through Figure 19. Here, we calculated the quotient of the bit error rate using one metric and 100 metrics. From $M \geq 6$ onwards, $\frac{\mathrm{BER}(1,2^M)}{\mathrm{BER}(100,2^M)}$ approaches ~1, which means, no real improvement is possible anymore through the S-Metric method.

Impact of helper data size

The amount of helper data bits required by SMHD is defined as a function of the number of metrics as $\log_2(S)$. The overall extracted-bits to helper-data-bits ratio can be defined here as $\mathscr{r} = \frac{M}{\log_2(S)}$

Table 5: S-Metric performance with same bit-to-metric ratios

M	1	2	3	4	5	6
S	2	4	8	16	32	64
BER	0.012	$0.9 \cdot 10^{-4}$	0.002	0.025	0.857	0.148

If we take a look at the error rates of configurations for which $\mathscr{r}$ is $800 \cdot 1$, we can observe a decline in performance of SMHD for general higher-bit quantization processes. This behaviour is also shown in Table 5.

Impact of temperature

We will now take a look at the impact on the error rates of changing the temperature both during the enrollment and the reconstruction phase.

The most common case to look at, is if we consider a fixed temperature during enrollment, most likely $25°C$. Since we wont always be able to recreate lab-like conditions during the reconstruction phase, it makes sense to look at the error rates at which reconstruction was performed at different temperatures.

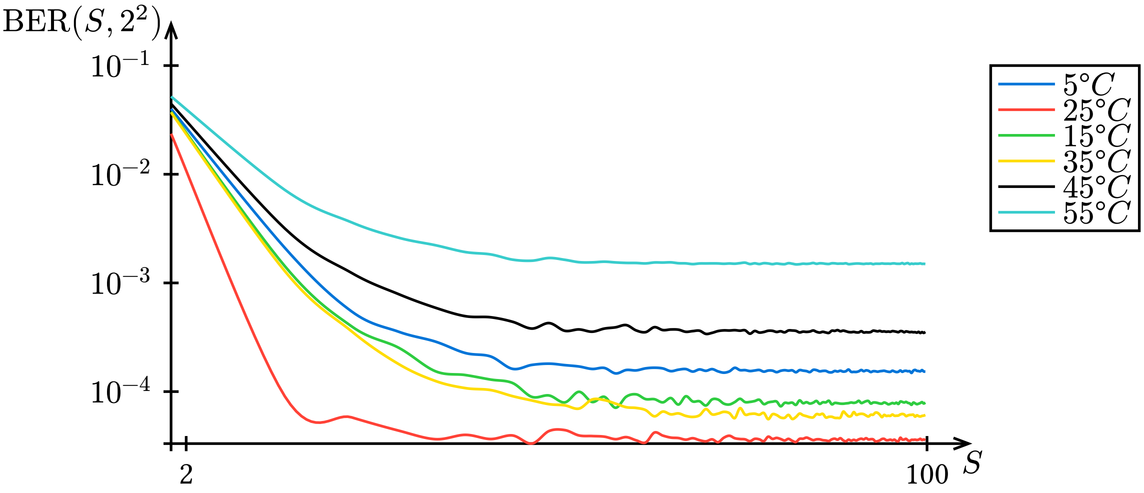


Figure 20: BERs for reconstruction at different temperatures. Generally, the further we move away from the enrollment temperature, the worse the BER gets.

Figure 20 shows the results of this experiment conducted with a 2-bit configuration.
As we can see, the further we move away from the temperature of enrollment, the higher the BERs. We can observe this property well in detail in Figure 21.

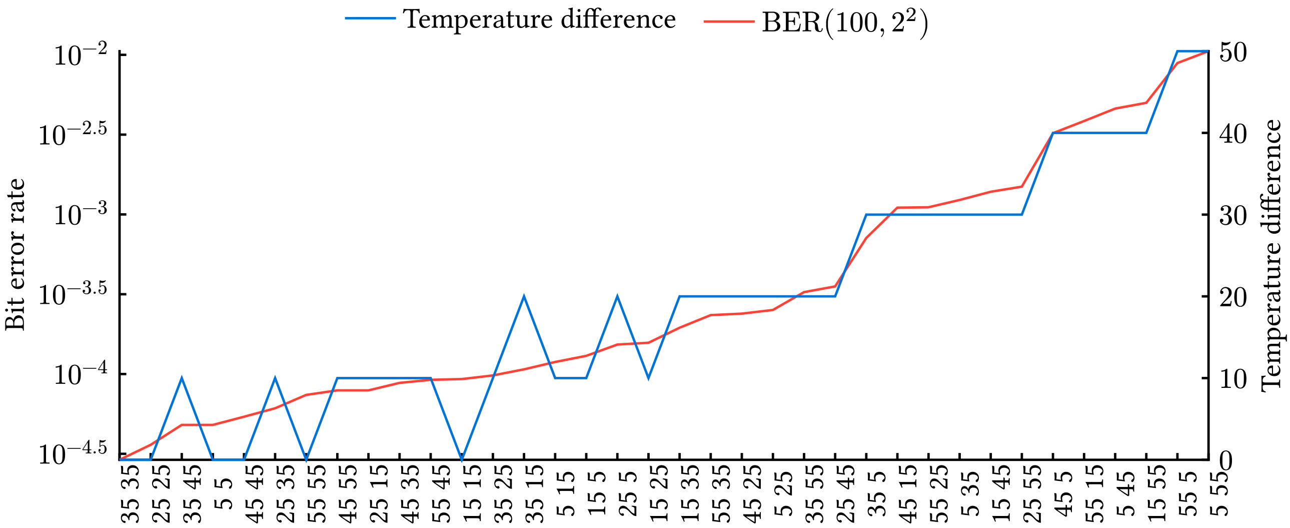


Figure 21: BERs for different enrollment and reconstruction temperatures. The lower number in the operating configuration is assigned to the enrollment phase, the upper one to the reconstruction phase. The correlation between the BER and the temperature is clearly visible here

Here, we compared the asymptotic performance of SMHD for different temperatures both during enrollment and reconstruction. First we can observe that the optimum temperature for the operation of SMHD in both phases for the dataset [12] is $35°C$ instead of the expected $25°C$. Furthermore, the BER seems to be almost directly determined by the absolute temperature difference, especially at higher temperature differences, showing that the further apart the temperatures of the two phases are, the higher the BER.

Gray coding

In Section 2.3, we discussed how a gray coded labelling for the quantizer could improve the bit error rates of the S-Metric method.

Because we only change the labelling of the quantizing bins and do not make any changes to SMHD itself, we can assume that the effects of temperature on the quantization process are directly translated to the gray-coded case.

Figure 22 shows the comparison of applying SMHD at room temperature for both naive and gray-coded labels. There we can already observe the improvement of using gray-coded labelling, but the impact of this change of labels can really be seen in Table 6. As we can see, the improvement rises rapidly to a peak at a bit width of M=3 and then falls again slightly. This effect can be explained with the exponential rise of the BER for higher bit widths M. For $M > 3$ the rise of the BER predominates the possible improvement by applying a gray-coded labelling.

Table 6: Improvement of using gray-coded instead of naive labelling, per bit width

1	2	3	4	5	6
0%	24.75%	47.45%	46.97%	45.91%	37.73%

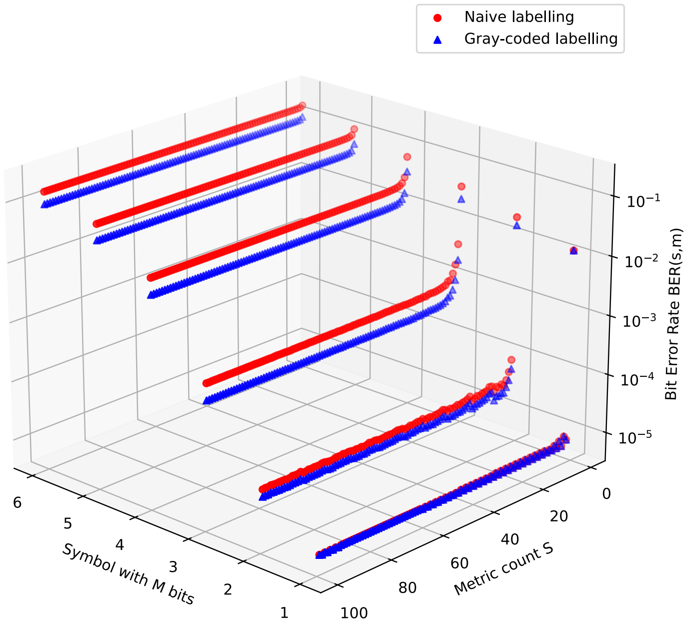


Figure 22: Comparison between BERs using naive labelling and gray-coded labelling

Using the dataset, we can estimate the average improvement for using gray-coded labelling to be at 33%.

3 Boundary Adaptive Clustering with Helper Data (BACH)

We can explore the option of finding helper data before performing the quantization process. This approach aims to optimize our input values prior to quantization, which may help minimize the risk of bit and symbol errors during the reconstruction phase. This differs from methods like SMHD, which generate helper data to improve the quantization process itself, of those that apply error-correcting codes afterward.

Since this HDA modifies the input values before the quantization takes place, we will consider the input values as zero-mean Gaussian distributed and not use a CDF to transform these values into the tilde-domain.

3.1 Optimizing single-bit sign-based quantization

Before we take a look at the higher order quantization cases, we will start with a very basic method of quantization: a quantizer, that only returns a symbol with a width of 1 bit and uses the sign of the input value to determine the resulting bit symbol.

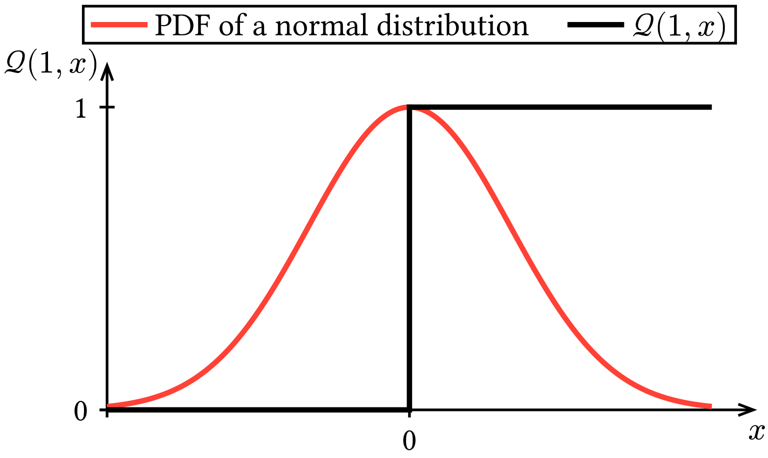


Figure 23: 1-bit quantizer with the PDF of a normal distribution

If we overlay the PDF of a zero-mean Gaussian distributed variable X with a sign-based quantizer function as shown in Figure 23, we can see that the expected value of the Gaussian distribution overlaps with the decision threshold of the sign-based quantizer. Considering that the margin of error of the value x is comparable with the one shown in Figure 4, we can conclude that values of X that reside near 0 are to be considered more unreliable than values that are further away from the x-value 0. This means that the quantizer used here is very unreliable as is.

Now, to increase the reliability of this quantizer, we can try to move our input values further away from the value $x = 0$. To do so, we can define a new input value z as a linear combination of two realizations of X, x_1 and x_2 with a set of weights h_1 and h_2 that we will use as helper data:

$$z = h_1 \cdot x_1 + h_2 \cdot x_2, \tag{17}$$

with $h_i \in \{\pm 1\}$. Building only the sum of two input values $x_1 + x_2$ is not sufficient here, since the resulting distribution would be a normal distribution with $\mu = 0$ as well.

3.1.1 Derivation of the resulting distribution

To find a description for the random distribution Z of z we can interpret this process mathematically as a maximisation of a sum. This can be realized by replacing the values of x_i with their absolute values as this always gives us the maximum value of the sum:

$$z = |x_1| + |x_2| \tag{18}$$

Taking into account that x_i are realizations of a normal distribution, we can assume without loss of generality that X is i.i.d., defining the overall resulting random distribution Z as:

$$Z = |X_1| + |X_2|. \tag{19}$$

We will redefine $|X|$ as a half-normal distribution Y whose PDF is

$$f_{Y(y,\sigma)} = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{y^2}{2\sigma^2}\right)\Bigg|_{\sigma=1}, y \geq 0 \tag{20.1}$$

$$= \sqrt{\frac{2}{\pi}} \exp\left(-\frac{y^2}{\sigma^2}\right). \tag{20.2}$$

Now, Z simplifies to

$$Z = Y_1 + Y_2. \tag{21}$$

We can assume for now that the realizations of Y are independent of each other. The PDF of the addition of these two distributions can be described through the convolution of their respective PDFs:

$$f_{Z(z)} = \int_0^z f_Y(y) f_Y(z-y) dy \tag{22.1}$$

$$= \int_0^z \left[\sqrt{\frac{2}{\pi}} \exp\left(-\frac{y^2}{2}\right) \sqrt{\frac{2}{\pi}} \exp\left(-\frac{(z-y)^2}{2}\right)\right] dy \tag{22.2}$$

$$= \frac{2}{\pi} \int_0^z \exp\left(-\frac{y^2 + (z-y)^2}{2}\right) dy \tag{22.3}$$

Evaluating the integral of Equation 22.3, we can now describe the resulting distribution of this maximisation process analytically:

$$f_Z = \frac{2}{\sqrt{\pi}} \exp\left(-\frac{z^2}{4}\right) \operatorname{erf}\left(\frac{z}{2}\right) z \geq 0. \tag{23}$$

Our derivation of f_Z currently only accounts for the addition of positive values of x_i, but two negative x_i values would also return the maximal distance to the coordinate origin. The derivation for the corresponding PDF is identical, except that the half-normal distribution Equation 20 is mirrored around the y-axis. Because the resulting PDF f_Z^{neg} is a mirrored variant of f_Z and f_Z is arranged symmetrically around the origin, we can define a new PDF f_Z^* as

$$f_Z^*(z) = |f_Z(z)|, \tag{24}$$

on the entire z-axis. $f_Z^*(z)$ now describes the final random distribution after the application of our optimization of the input values x_i.

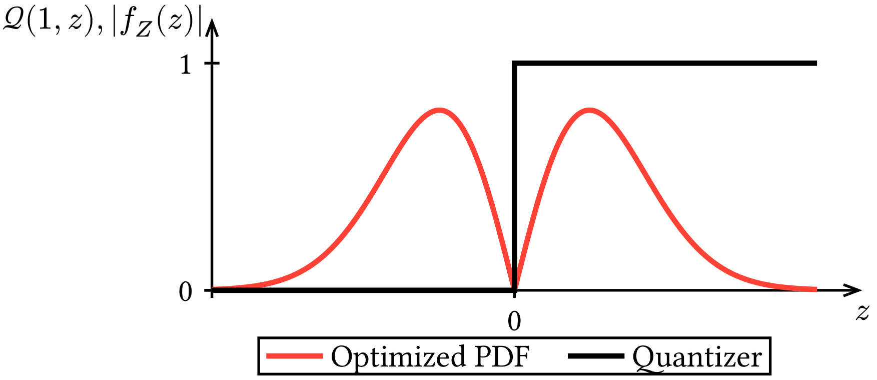


Figure 24: Optimized input values z overlaid with sign-based quantizer $\mathcal{Q}$

Figure 24 shows two key properties of this optimization:

1. Adjusting the input values using the method described above does not require any adjustment of the decision threshold of the sign-based quantizer.
2. The resulting PDF is zero at $z = 0$ leaving no input value for the sign-based quantizer at its decision threshold.

3.1.2 Generating helper-data

To find the optimal set of helper-data that will result in the distribution shown in Figure 24, we can define the vector of all possible linear combinations $\boldsymbol{z}$ as the vector-matrix multiplication of the input values x_i and the matrix $\boldsymbol{H}$ of all weight combinations with $h_i \in [\pm 1]$:

$$\boldsymbol{z} = \boldsymbol{x} \cdot \boldsymbol{H} \tag{25}$$

We will choose the optimal weights based on the highest absolute value of $\boldsymbol{z}$, as that value will be the furthest away from 0. To not encounter two entries in $\boldsymbol{z}$ that both have the same highest absolute value, we can set the first helper data bit to be always $h_1 = 1$.

Considering our single-bit quantization case, Equation 25 can be written as:

$$\boldsymbol{z} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot \begin{bmatrix} +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \end{bmatrix} \tag{26}$$

The vector of optimal weights $\boldsymbol{h}_{\text{opt}}$ can now be found through $\text{argmax}_h(\boldsymbol{z})$. If we take a look at the dimensionality of the matrix of all weight combinations, we notice that we will need to store only 1 helper-data bit per quantized symbol because h_1 is set to 1. In fact, we will show later, that the amount of helper-data bits used by this HDA is directly linked to the number of input values used instead of the number of bits we want to extract during quantization.

3.2 Generalization to higher-order bit quantization

We can generalize the idea of Section 3.1 and apply it for a higher-order bit quantization. Contrary to SMHD, we will always use the same step function as quantizer and optimize the input values x to be the furthest away from any decision threshold. In this higher-order case, this means that we want to optimise our input values as far away as possible from the nearest decision threshold of the quantizer instead of just maximising the absolute value of the linear combination.

For a complete generalization of this method, we will also parametrize the amount of addends N kin the linear combination of z. That means we can define z from now on as:

$$z = \sum_{i=1}^{N} x_i \cdot h_i \tag{27}$$

We can define the condition to test whereas a tested linear combination is optimal as follows:
The optimal linear combination z_{opt} is found, when the distance to the nearest quantizer decision bound is maximised. Finding the weights $\boldsymbol{h}_{\text{opt}}$ of the optimal linear combination z_{opt} can be formalized as:

$$\boldsymbol{h}_{\text{opt}} = \text{argmax}_h \ \text{min}_j \left|\boldsymbol{h}^T \boldsymbol{x} - b_j\right| \ \text{ s.t. } h_j \in \{\pm 1\} \tag{28}$$

Example with 2-bit quantizer

We can define the bounds of the two bit quantizer $\boldsymbol{b}$ as $[-\alpha, 0, \alpha]$ omitting the bounds $\pm\infty$. The values of $\boldsymbol{b}$ are already placed in the real domain to directly quantize normal distributed input values. A simple way to solve Equation 28 is to use a brute force method and calculate all distances to every quantization bound b_j, because the number of possible combinations is finite. Furthermore, fining a solution for Equation 28 analytically poses to be significantly more complex.

The linear combination z for the amount of addends $i = 2$ is defined as

$$z = x_1 \cdot h_1 + x_2 \cdot h_2 \tag{29}$$

According to Equation 25, all possible linear combinations for two input values x_1 and x_2 of Equation 29 can be collected as the vector $\boldsymbol{z}$ of length $2^i \ |_{i=2} = 4$:

$$\boldsymbol{z} = (z_1, z_2, z_3, z_4) \tag{30}$$

Calculating the absolute distances to every quantizer bound b_i for all linear combinations z_i gives us the following distance matrix:

$$\boldsymbol{\mathcal{A}} = \begin{pmatrix} a_{1,1} & a_{2,1} & a_{3,1} & a_{4,1} \\ a_{1,2} & a_{2,2} & a_{3,1} & a_{4,2} \\ a_{1,3} & a_{2,3} & a_{3,1} & a_{4,3} \end{pmatrix}, \tag{31}$$

where $a_{\mathrm{i,j}} = |z_i - b_j|$.

Now we want to find the bound b_i for every z_i to which it is closest. This can be achieved by determining the minimum value for each column of the matrix $\boldsymbol{\mathcal{A}}$. The resulting vector $\boldsymbol{\nu}$ now consists of the distance to the nearest quantizer bound for every linear combination with entries defined as:

$$\nu_j = \min\{a_{\mathrm{i,j}} \mid 1 \leq j \leq 4\} \text{ for each } i = 1, 2, 3. \tag{32}$$

The optimal linear combination z_{opt} can now be found as the entry z_j of $\boldsymbol{z}$ where its corresponding distance ν_j is maximised.

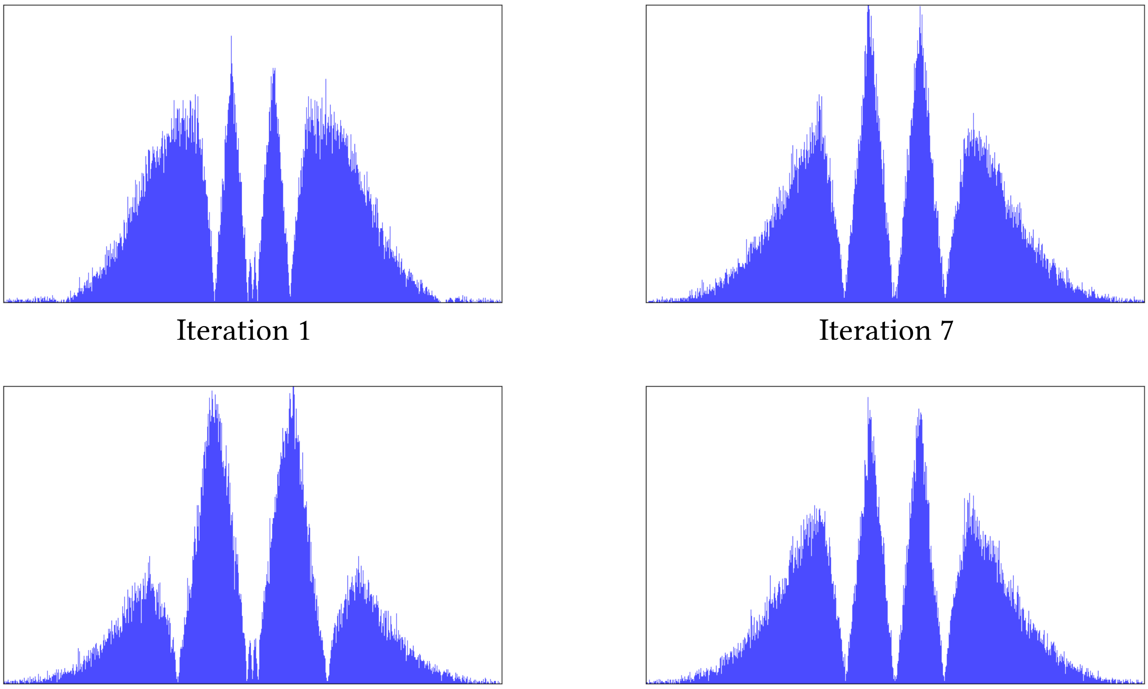
3.2.2 Simulation of the bound distance maximisation strategy

Two important points were anticipated in the preceding example:

1. We cannot define the resulting random distribution Z after performing this operation analytically and thus also not the quantizer bounds $\boldsymbol{b}$. A way to account for that is to guess the resulting random distribution and $\boldsymbol{b}$ initially and repeating the optimization using quantizer bounds found through the eCDF of the resulting linear combination values.
2. If the optimization described above is repeated multiple times using an eCDF, the resulting random distribution Z must converge to a stable random distribution. Otherwise we will not be able to carry out a reliable quantization in which the symbols are uniformly distributed.

To check that the strategy for optimizing the linear combination provided in the example above results in a converging random distribution, we will perform a simulation of the optimization as described in the example using 100 000 simulated normal distributed values as realizations of the standard normal distribution with the parameters $\mu = 0$ and $\sigma = 1$.

Figure 25 shows various histograms of the vector $\boldsymbol{z}_{\mathrm{opt}}$ after different iterations. Even though the overall shape of the distribution comes close to our goal of moving the input values away from the quantizer bounds $\boldsymbol{b}$, the distribution itself does not converge to one specific, final shape. It seems that the resulting distributions for each iteration oscillate in some way, since the distributions for iterations 7 and 25 have the same shape. However the distribution seems to be chaotic and thus does not seem suitable for further quantization.



Iteration 1

Iteration 7

Iteration 18

Iteration 25

Figure 25: Probability distributions for various iterations

3.2.3 Center Point Approximation

For that reason, we will now propose a different strategy to find the weights for the optimal linear combination z_{opt}. Instead of defining the desired outcome of z_{opt} as the greatest distance to the nearest quantizer decision threshold, we will define a vector $\boldsymbol{o} = [o_1, o_2 ..., o_{2^M}]$ containing the optimal values that we want to approximate with z. Considering a M-bit quantizer with 2^M steps, we can define the values of $\boldsymbol{o}$ as the center points of these quantizer steps. Its cardinality is 2^M. It has to be noted, that $\boldsymbol{o}$ consists of optimal values that we may not be able to exactly approximate using a linear combination based on weights and our given input values.

We can find the optimal linear combination z_{opt} by finding the minimum of all distances to all optimal points defined in $\boldsymbol{o}$. The matrix that contains the distances of all linear combinations $\boldsymbol{z}$ to all optimal points $\boldsymbol{o}$ is defined as: $\boldsymbol{\mathcal{A}}$ with its entries $a_{\text{ij}} = |z_i - o_j|$.
z_{opt} can now be defined as the minimal value in $\boldsymbol{\mathcal{A}}$:

$$z_{\text{opt}} = \min(\boldsymbol{\mathcal{A}}) = \min\left(\begin{bmatrix} a_{00} & \cdots & a_{\text{i0}} \\ \vdots & \ddots & \\ a_{\text{0j}} & & a_{\text{ij}} \end{bmatrix}\right). \tag{33}$$

Algorithm 2: OptimalWeights to approximate $\boldsymbol{\mathcal{o}}$

```
inputs:
    $\boldsymbol{y}$ input values for linear combinations
    $\boldsymbol{\mathcal{o}}$ list of optimal points
output: $(\boldsymbol{h}, z_{\text{opt}})$
calculate all possible linear combinations $\boldsymbol{z}$ with Equation 27
calculate matrix $\boldsymbol{\mathcal{A}}$ with $a_{\text{ij}} = |z_i - \boldsymbol{\mathcal{o}}_j|$
return weights $\boldsymbol{h}$ for $z_{\text{opt}} = \text{argmin}(\boldsymbol{\mathcal{A}})$ and $z_{\text{opt}}$
```

Algorithm 2 shows a programmatic approach to find the set of weights for the best approximation. The algorithm returns a tuple consisting of the weight combination $\boldsymbol{h}$ and the resulting value of the linear combination z_{opt}.

Because the superposition of different linear combinations of normal distributions corresponds to a Gaussian Mixture Model, finding the ideal set of points $\boldsymbol{\mathcal{o}}$ analytically is impossible.

Instead, we will first estimate $\boldsymbol{\mathcal{o}}$ based on the normal distribution parameters after performing multiple convolutions with the input distribution X. The parameters of a multiple convoluted normal distribution is defined as:

$$\sum_{i=1}^{n} \mathcal{N}(\mu_i, \sigma_i^2) \sim \mathcal{N}\left(\sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2\right), \tag{34}$$

while n defines the number of convolutions performed [13].

With this definition, we can define the parameters of the probability distribution Z of the linear combinations z based on the parameters of X, μ_X and σ_X:

$$Z(\mu_Z, \sigma_Z^2) = Z\left(\sum_{i=1^n} \mu_X, \sum_{i=1}^{n} \sigma_X^2\right) \tag{35}$$

The parameters μ_Z and σ_Z allow us to apply an inverse CDF on a multi-bit quantizer $\mathcal{Q}(2, \tilde{x})$ defined in the tilde-domain. Our initial values for $\boldsymbol{\mathcal{o}}_{\text{first}}$ can now be defined as the centers of the steps of the transformed quantizer function $\mathcal{Q}(2, x)$. These points can be found easily but for the outermost center points whose quantizer steps have a bound $\pm\infty$.
However, we can still find these two remaining center points by artificially defining the outermost bounds of the quantizer as $\frac{1}{2^{2 \cdot M}}$ and $\frac{(2^{2 \cdot M}) - 1}{2^{2 \cdot M}}$ in the tilde-domain and also apply the inverse CDF to them.

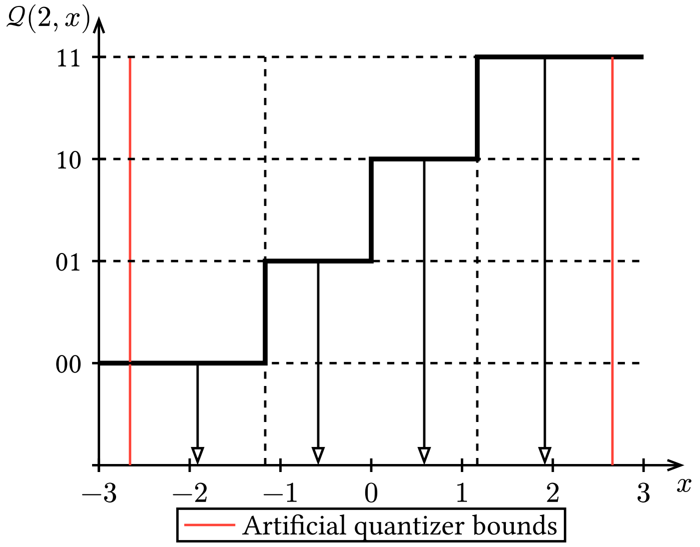


Figure 26: Quantizer for the distribution resulting a triple convolution with distribution parameters $\mu_X = 0$ and $\sigma_X = 1$ with marked center points of the quantizer steps

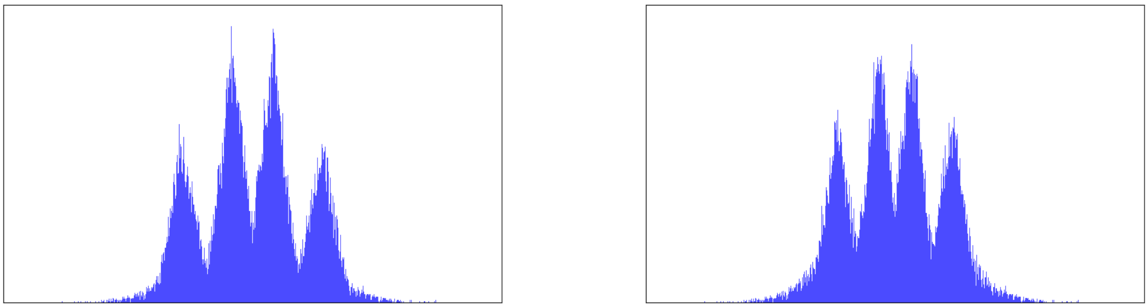
We can now use an iterative algorithm that alternates between optimizing the quantizing bounds of $\mathcal{Q}$ and our vector of optimal points $\boldsymbol{o}_{\text{first}}$.

Algorithm 3: Center Point Approximation

```
1  input: $\boldsymbol{o}_{\text{first}}, \boldsymbol{x}, t, M$
2  lists: optimal weights $\boldsymbol{h}_{\text{opt}}$
3  $\boldsymbol{o} \leftarrow \boldsymbol{o}_{\text{first}}$
4  repeat t times:
5      perform OptimalWeights$(0, \boldsymbol{x})$:
6          update $\boldsymbol{h}_{\text{opt}}$ with returned weights
7          $\boldsymbol{z}_{\text{opt}} \leftarrow$ all returned linear combinations
8      define new quantizer $\mathcal{Q}^*$ using the eCDF based on $\boldsymbol{z}_{\text{opt}}$:
9          sort $\boldsymbol{z}_{\text{opt}}$ in ascending order
10         $\mathcal{Q}^* \leftarrow$ use Equation 4 with quantizer bounds in the tilde domain
11     update $\boldsymbol{o}$ with newly found quantizer step centers
12 return $\boldsymbol{h}_{\text{opt}}$
```

We can see both of these alternating parts in Lines 6 and 11 of Algorithm 3. To optimize the quantizing bounds of $\mathcal{Q}$, we will sort the values of all the resulting linear combinations $\boldsymbol{z}_{\text{opt}}$ in ascending order. Using the inverse eCDF defined in Equation 4, we can find new quantizer bounds based on $\boldsymbol{z}_{\text{opt}}$ from the first iteration. These bounds will then be used to define a new set of optimal points $\boldsymbol{o}$ used for the next iteration. During every iteration of Algorithm 3, we will store all weights $\boldsymbol{h}$ used to generate the vector for optimal linear combinations $\boldsymbol{z}_{\text{opt}}$.

We can also use a simulation here to check the convergence of the distribution Z using the same input values and quantizer configurations as in Section 3.2.2.



Iteration 1

Iteration 25

Figure 27: Probability distributions for the first and 25th iteration of the center point approximation method

Comparing the distributions in Figure 27, we can see that besides a closer arrangement the overall shape of the probability distribution Z converges to a stable distribution representing the original estimated distribution Z through Equation 35 through smaller normal distributions.

The output of Algorithm 3 is the vector of optimal weights $\boldsymbol{h}_{\text{opt}}$. $\boldsymbol{h}_{\text{opt}}$ can now be used to complete the enrollment phase and quantize the values $\boldsymbol{z}_{\text{opt}}$.

To perform reconstruction, we can calculate the same linear combination used during enrollment with the generated helper-data and the new PUF readout measurements. We can lower the computational complexity of this approach by using the assumption that X are i.i.d.. The end result of $\boldsymbol{o}$ can be calculated once for a specific device series and saved in the ROM of. During enrollment, only the vector $\boldsymbol{h}_{\text{opt}}$ has to be calculated.

3.2.4 Impact of helper-data volume and amount of addends

The amount of helper data is directly linked to the symbol bit width M and the amount of addends N used in the linear combination. Because we can set the first helper data bit h_1 of a linear combination to 1 to omit the random choice, the resulting extracted bit to helper data bit ratio r can be defined as $r = \frac{M}{N-1}$, whose equation is similar tot he one we used in the SMHD analysis.

3.3 Experiments

To test our implementation of BACH using the prior introduced center point approximation we conducted a similar experiment as in Section 2.4. However, we have omitted the analysis over different temperatures for the enrollment and reconstruction phase here, as the behaviour of BACH corresponds to that of SMHD in this matter. As in the S-Metric analysis, the resulting dataset consists of the bit error rates of various configurations with quantization symbol widths of up to 4 bits evaluated with up to 10 addends for the linear combinations.

3.4 Results & Discussion

We can now compare the BERs of different BACH configurations.

Table 7: BERs of different BACH configurations

BER	N=2	N=3	N=4	N=5	N=6	$N = 7$	$N = 8$	$N = 9$
$M = 1$	**0.01**	**0.01**	**0.012**	**0.018**	**0.044**	**0.05**	**0.06**	**0.07**
$M = 2$	**0.03**	**0.05**	**0.02**	**0.078**	**0.107**	**0.114**	**0.143**	**0.138**
$M = 3$	**0.07**	**0.114**	**0.05**	**0.15**	**0.2**	**0.26**	**0.26**	**0.31**
$M = 4$	**0.13**	**0.09**	**0.18**	**0.22**	**0.26**	**0.31**	**0.32**	**0.35**
$M = 5$	**0.29**	**0.21**	**0.37**	**0.31**	**0.23**	**0.23**	**0.19**	**0.15**
$M = 6$	**0.15**	**0.33**	**0.15**	**0.25**	**0.21**	**0.23**	**0.19**	**0.14**

Table 7 shows the BERs of BACH configurations with N addends and extracting M bits out of one input value z. The first interesting property we can observe, is the caveat BACH produces for the first three bit combinations $M = 1, 2$ and 3 at around $N = 3$ and $N = 4$. At these points, the BER experiences a drop followed by a steady rise again for higher numbers of N. If M is generally chosen higher, BACH seems to return unstable results, halving the BER as N reaches 9 for $M = 5$ but showing no real improvement for various addends if $M = 6$.

We can also compare the performance of BACH using the center point approximation approach with the BERs of higher order bit quantizations that don't use any helper data.

Table 8: BERs for higher order bit quantization without helper data

M	1	2	3	4	5	6
BER	0.013	0.02	0.04	0.07	0.11	0.16

Unfortunately, the comparison of BERs of Tables 8 and 7 shows that our current realization of BACH does either ties the BER in Table 8 or is worse. Let's find out why this happens.

Justification of the original idea

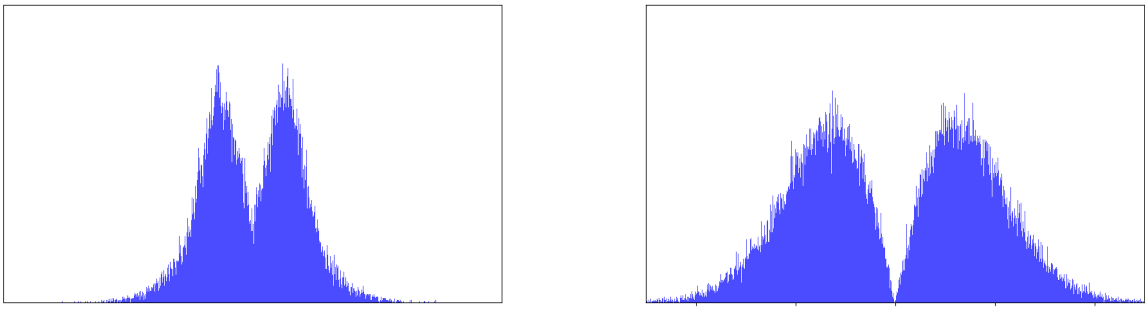
If we take a step back and look at the performance of the optimized single-bit sign-based quantization process of Section 3.1, we can compare the following BERs:

Table 9: Comparison of BERs for the single-bit quantization process with and without helper data

No helper data	0.013
With helper data using greatest distance	0.00052
With helper data using center point approximation	0.01

As we can see in Table 9, generating the helper data based on the original idea where Equation 28 is used improves the BER of the single-bit quantization by approx. 96%. The probability distributions

Z of the two different realizations of BACH – namely the distance maximization strategy and the center point approximation – give an indication of this discrepancy:



Center point approximation

Distance maximization

Figure 30: Comparison of the histograms of the different strategies to obtain the optimal weights for the single-bit case

Figure 30 shows the two different probability distributions. We can observe that using a vector of optimal points $\boldsymbol{o}$ results in a more narrow distribution for Z than just maximizing the linear combination to be as far away from $x = 0$ as possible. This difference in the shape of both distributions seem to be the main contributor to the fact that the optimization using center point approximation yields no improvement for the quantization process. Unfortunately, we were not able define an algorithm translating this idea to a higher order bit quantization for which the resulting probability distribution Z converges.

Taking a look at the unstable probability distributions issued by the bound distance maximization strategy in Figure 25, we can get an idea of what kind of distribution a BACH algorithm should achieve. While the inner parts of the distributions do not overlap with each other like in the stable iterations shown in Figure 27, the outermost values of these distributions resemble the shape of what we achieved using the distance maximization for a single-bit optimization. These two properties could – if the distribution converges – result in far better BERs for higher order bit quantization, as the comparison in Table 9 indicates.

4 Conclusion and Outlook

During the course of this work, we took a closer look at an already introduced HDA, SMHD and provided a concrete realization. Our experiments showed that after a certain point, using more metrics S won't improve the BER any further as they behave asymptotically for $S \to \infty$. Furthermore, we concluded that for higher choices of the symbol width M, SMHD will not be able to improve on the BER, as the initial error is too high. An interesting addition to our analysis provided the improvement of Gray-coded labelling for the quantizer as this resulted in an improvement of $\approx 30\%$.

Going on, we introduced the idea of a new HDA which we called Boundary Adaptive Clustering with Helper data BACH. Here we aimed to utilize the idea of moving our initial PUF measurement values away from the quantizer bound to reduce the BER using weighted linear combinations of our input values. Although this method posed promising results for a sign-based quantization yielding an improvement of $\approx 96\%$ in our testing, finding a good approach to generalize this concept turned out to be difficult. The first issue was the lack of an analytical description of the probability distribution resulting from the linear combinations. We accounted for that by using an algorithm that alternates between defining the quantizing bounds using an eCDF and optimizing the weights for the linear combinations based on the found bounds. The loose definition of Equation 28 to find an ideal linear combination which maximizes the distance to its nearest quantization bound did not result in a stable probability distribution over various iterations. Thus, we proposed a different approach to approximate the linear combination to the centers between the quantizing bounds. This method resulted in a stable probability distribution, but did not provide any meaningful improvements to the BER in comparison to not using any helper data at all.

Future investigations of the BACH idea might find a solution to the convergence of the bound distance maximization strategy. Since the vector of bounds $\boldsymbol{b}$ is updated every iteration of BACH, a limit to the deviation from the previous position of a bound might be set. Furthermore, a recursive approach to reach higher order bit quantization inputs might also result in a converging distribution. If we do not want to give up the approach using a vector of optimal points $\boldsymbol{o}$ as in the center point approximation, a way may be found to increase the distance between all optimal points $\boldsymbol{o}$ to achieve a better separation for the results of the linear combinations in every quantizer bin.

If a converging realization of BACH is found, using fractional weights instead of ± 1 could provide more flexibility for the outcome of the linear combinations.

Ultimately, we can build on this in the future and provide a complete key storage system using BACH or SMHD to improve the quantization process.

But in the end, the real quantizers were the friends we made along the way.

Glossary

***BACH* – Boundary Adaptive Clustering with Helper data**. 8, 9, 33, 34, 35, 37

***BER* – bit error rate**. 19, 20, 21, 22, 23, 34, 37

***CDF* – cumulative distribution function**. 9, 13

***eCDF* – empirical Cumulative Distribution Function**. 5, 9, 29, 32, 37

***HDA* – helper data algorithm**. 7, 8, 11, 25, 37

***PUF* – physical unclonable function**. 7, 8, 11, 13, 37

***SMHD* – S-Metric Helper Data method**. 5, 8, 9, 12, 13, 20, 21, 22, 25, 28, 33, 37

***TMHD* – Two Metric Helper Data method**. 7, 8, 11

Bibliography

[1] M. Garcia-Bosque, G. D\iez-Señorans, C. Sánchez-Azqueta, and S. Celma, "Introduction to physically unclonable fuctions: Properties and applications," in *2020 European Conference on Circuit Theory and Design (ECCTD)*, 2020, pp. 1–4.

[2] R. Maes, "Physically Unclonable Functions: Constructions, Properties and Applications (Fysisch onkloonbare functies: constructies, eigenschappen en toepassingen)," 2012.

[3] V. Immler, M. Hiller, Q. Liu, A. Lenz, and A. Wachter-Zeh, "Variable-length bit mapping and error-correcting codes for higher-order alphabet pufs—extended version," *Journal of Hardware and Systems Security*, vol. 3, pp. 78–93, 2019.

[4] T. Ignatenko and F. Willems, "Achieving secure fuzzy commitment scheme for optical pufs," in *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2009, pp. 1185–1188.

[5] J. Ruchti, M. Gruber, and M. Pehl, "When the Decoder Has to Look Twice: Glitching a PUF Error Correction," *Cryptology ePrint Archive*, 2021.

[6] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2014.

[7] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *2009 IEEE international symposium on information theory*, 2009, pp. 2101–2105.

[8] J.-L. Danger, S. Guilley, and A. Schaub, "Two-metric helper data for highly robust and secure delay PUFs," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (I-WASI)*, 2019, pp. 184–188.

[9] L. Tebelmann, U. Kühne, J.-L. Danger, and M. Pehl, "Analysis and protection of the two-metric helper data scheme," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, 2021, pp. 279–302.

[10] R. F. Fischer, "Helper Data Schemes for Coded Modulation and Shaping in Physical Unclonable Functions," *arXiv preprint arXiv:2402.18980*, 2024.

[11] F. M. Dekking, *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[12] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, "Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 126–133.

[13] R. V. Hogg, J. W. McKean, A. T. Craig, and others, *Introduction to mathematical statistics*. Pearson Education India, 2013.