Volume 122, Article No. 42 (2017) https://doi.org/10.6028/jres.122.042
Journal of Research of the National Institute of Standards and Technology

Polar Codes for Quantum Key Distribution

Anastase Nakassis

National Institute of Standards and Technology,
Gaithersburg, MD 20899, USA

anakassis@nist.gov

This paper addresses the performance of polar codes in the context of the quantum key distribution (QKD) protocol. It introduces the notion of an interactive polar decoder and studies its performance. The results demonstrate that the interactive decoder is efficient in the right environment and can be used to construct good classical polar codes efficiently.

Key words: frozen bits; polar codes; quantum key distribution (QKD); reconciliation.

Accepted: September 28, 2017

Published: December 14, 2017

https://doi.org/10.6028/jres.122.042

1. Introduction

The polar codes are a class of linear block error-correcting codes transmitted over symmetric binary-input discrete memoryless channels [1, 2, 3]. As their length N, $N = 2^n$, increases, their performance tends to the Shannon limit. Subsequent publications (e.g., [4, 5]) have addressed practical aspects of the polar codes such as the size length, N, of a polar code that can support a specific feasible performance profile and the impact of the available arithmetic precision on the performance of the polar decoder.

As shown in Refs. [6, 7], the polar codes can be used in the reconciliation stage of the quantum key distribution (QKD) protocol. The QKD protocol [8] creates shared secrets by using a quantum channel for “data” that suffers massive deletions (50 % or more) and high bit-error rates (typically between 1 % and 4%, and in theory as high as 11%) and it resolves the bit-value discrepancies through information exchanged over a classical channel that supports data integrity, origin authentication, and protection against replays. The first sound error-correcting protocol to be used by QKD, Cascade [9, 10], is interactive.

Cascade went out of fashion because it was believed that it has latency problems. As a result, the use of other, noninteractive decoding schemes, such as the “Low-density parity-check”, LDPC, code [11–13] and polar codes [1–7] were proposed. It should be noted, however, that Cascade is performing a return [14, 15] and there are claims [15] that Cascade currently has no real latency problems.

The objective of the QKD protocol’s reconciliation stage is to correct errors in the quantum channel data in such a way that the expected secrecy yield is as high as possible. This translates into maximizing the following:

$$(1 - \text{FER})\times(K - N \times h(QBER) - \text{signature\_length}.$$

where K is the number of information bits in the code, $QBER$ is the estimated error rate in the quantum channel, FER is the frame error rate (*i.e.*, the probability of erroneous decoding), signature_length is the length, in bits, of a hash value used to detect incorrect decoding, and $h(QBER)$ is the Shannon entropy.

This paper introduces an interactive polar decoder, studies the performance of such a decoder, and shows that such a decoder can efficiently produce good classical polar codes.

It should be noted that the presented simulation data hold for the QKD protocol and not necessarily for its implementations; the latter must adjust to changes to their environment and parameter values that are variable over time and never fully known.

2. Polar Coding/Decoding Summary

The Polar coding/decoding has been described in detail in several papers [1, 2, 5] and we shall assume basic familiarity with the encoding/decoding setup of polar codes. Nevertheless, we do provide an appendix that presents a description of the polar transform and we summarize the parts of the polar encoding and decoding processes that are directly linked to the error correction in general, and in particular to the alternatives studied herein.

2.1 The Encoder

The encoder accepts a bit-string, **u**, of length N, $N = 2^n$, the values of which are stored in the N entries of the 0th column of a $N \times (n + 1)$ matrix (this paper follows the programming convention that all indices start with the value 0). The following hold for the encoding matrix and its contents:

- There is a subset of $\{0, 1,\dots,N - 1\}$, known as the frozen bits, such that the values of **u** over the frozen-bits are known prior to decoding. The performance of the polar decoding depends on the choice of the frozen bits' locations, not on their bit values.
- For $m = 0, 1, 2, \dots, n$, the encoder views the mth column as a sequence of 2^m segments of $2^{(n-m)}$ bit slots.
- If (X, Y) are the slots $(2s, 2s+1)$ of column m and the slots $(2t, 2t + 1)$ of the kth segment, where $k = 0, 1,\dots,2^{(n-m)} - 1$, in column m, then $2s = k2^{(n-m)} + 2t$.
  - o The polar encoder will store bit[X]$\oplus$ bit[Y] and bit[Y] in slots X^* and Y^* of the $(m + 1)$th column, with $X^* = k2^{(n-m)} + t$ and $Y^* = X^* + 2^{(n-m-1)}$.
  - o If X, Y, X^*, and Y^* are as above, knowledge of the location of any one of them fully defines the quadruple (X, Y, X^*, Y^*).

The following property of the polar codes is critical for the polar decoder:

If **u*** is the bit-vector in column n (*i.e.*, **u*** is the polar encoding of **u**), then

- bit[X*] and bit[Y*] are the parities of **u*** over nonintersecting subsets of $\{0,1,2,\dots,N - 1\}$, S[X*] and S[Y*], which have the same cardinality.
- In the absence of side information, bit[X*] and bit[Y*] are independent variables (hence the *f*-transform below).
- S[X] = is the union of S[X*] and S[Y*], and S[Y] = S[Y*].
- If a value has been assigned to bit[X], bit[X*] and bit[Y*] cease to be independent (hence the *g*-transform below).

2.2 The Decoder

The decoder creates an N × (n+1) matrix, the entries of which can hold a real number; the probability, or an equivalent parameter, that the corresponding slot of the encoder holds the value 0. What follows shows that unless one is careful, the computer may compute probabilities that convey little information or even none. The polar decoder functions as follows:

1. If p is the estimated *QBER* value, **u*** is the bit vector sent, and **v*** is the bit vector received, the probability that **u***$[i] = 0$, $p[n, i]$, equals $1 - p$ if **v***$[i] = 0$ and p otherwise.
2. When the probabilities $p[X^*]$ and $p[Y^*]$ have been computed, the decoder will, unless side information is available, compute $p[X]$ through the *f*-transform:

$$p[X] = f(p[X^*],p[Y^*]) = p[X^*]p[Y^*] + (1 - p[X^*])(1 - p[Y^*]). \tag{1}$$

3. When probabilities $p[X^*]$ and $p[Y^*]$ have been computed and a bit value has been assigned to entry X, the decoder will, unless additional side information is available, compute $p(\text{bit}[Y] = 0|\text{bit}[X])$ through the *g*-transform as follows:

$$\text{If } (\text{bit}[X] = 0),\ p[Y] = p[X^*]\times p[Y^*]/f(p[X^*],p[Y^*]). \tag{1a}$$

$$\text{If } (\text{bit}[X] = 1),\ p[Y] = (1 - p[X^*])\times p[Y^*]/(1 - f(p[X^*],p[Y^*])). \tag{1b}$$

2.3 The *r* Parameter

In the following the parameter $r[Z] = 1\text{-}2\times p(Z = 0)$ (hence $p(Z = 0) = (1 + r[Z])/2$) is used because this parameter shows more clearly how and why some computed probabilities tend to cluster around 0.5 (i.e., the corresponding *r* value is close to 0). In terms of the *r* parameter, the formulae (1), (1a), and (1b) become

$$r[X] = r[X^*]\times r[Y^*], \tag{2}$$

$$r(Y|X = 0) = (r[X^*] + r[Y^*])/(1 + r[X^*]\times r[Y^*]) \tag{2a}$$

$$\mathrm{r}(Y|X = 1) = (r[Y^*] - r[X^*])/(1 - r[X^*]r[Y^*]). \tag{2b}$$

Note: In the rare instance that the product $r[X^*]r[Y^*]$ is equal to 1 or −1, the values of X^* and Y^* are known, and the same holds for *X* and *Y*.

If $r = 1– 2\times\text{QBER}$, the probabilities assigned to the entries of column *n* are $(1 + r)/2$, if entries **v***[i] = 0, and $(1 - r)/2$ otherwise. In the following, unless otherwise stated, it is assumed that the decoder works with the *r* values.

Note that:

1. If slot X is in column $n - \mathrm{m}$, m = 1, 2,…,n and p[X] is computed exclusively through f-operations, p[X] will be equal either to $(1 + \mathrm{r}^{\mathrm{M}})/2$ or to $(1 - \mathrm{r}^{\mathrm{M}})/2$ with $\mathrm{M} = 2^m$.
2. If a bit value has been assigned to X in column $n - m$, and the probabilities p[X*] and p[Y*] have been computed exclusively through f-operations, then the g-operation for Y will return the value 0.5 with probability exceeding $(1 - \mathrm{u}^{\mathrm{M}})/2$. Indeed, either $r[X^*] = r[Y^*]$, with probability $(1 + r^M)/2$, or $r[X^*] + \mathrm{r}[Y^*] = 0$, with probability $(1 - \mathrm{r}^{\mathrm{M}})/2$). As a result,
   - o if $\mathrm{r}[X^*] = \mathrm{r}[Y^*]$ and bit[X] = 1, then $\mathrm{r}(Y|X = 1) = 0$, and
   - o if $\mathrm{r}[X^*] + r[Y^*] = 0$ and bit[X] = 0, then $\mathrm{r}(Y|X = 0) = 0$

   Eqs. (2, 2a, 2b).

The r[X] value’s collapse to 0 is an artifact of inadequate precision and can be avoided if sufficient precision is available, typically by software libraries up to the task. The collapse of r[Y] to zero can be either an artifact of inadequate precision or a true value.

A way to bypass these situations, if the resultant latency is accepted, is to use an interactive decoder that will query the encoder for the corresponding bit value whenever a computed *r* value falls within the interval $(-\delta/2,\ \delta/2)$ with an appropriately small value for δ.

2.4 Interactive and Proximate QKD Polar Decoders

As shown in Ref. [7], the QKD polar decoders can, through sampling, collect data and mimic the classical polar encoder. Given that the native computer operations and number representations have limited precision, two polar-proximate algorithms were investigated herein. The first, P_D in what follows, uses limited precision versions of the *f* and *g* functions. The second, I_D in what follows, is an interactive version of the polar decoder.

The limited-precision version is based on the idea that the *f* and *g* operations will never be allowed to return (directly or indirectly) an *r* value that falls within the interval $(-\delta, \delta)$. Whenever the *r* value falls within $(-\delta, \delta)$, the decoder will set *r* value= −δ, if the computed *r* value is negative; otherwise, it will set *r* value = δ.

The I_D version is based on the following ideas:

1. If the absolute value of the computed *r* value is sufficiently small, the decoder can ask that the corresponding cell’s bit value of the encoder be released. Sufficiently small in the following will be mean that the *r* value falls within $(-\delta, \delta)$; in what follows, and for reasons listed in Section 4 $\delta^2 \le 1/N$.
2. If the decoder has computed *p*[*k*], the probability that the bit value in the *k*th slot of column 0 is 0, and *p*[*k*] is sufficiently close to 0.0 or to 1.0 we use *p*[*k*] to assign a value to the *k*th slot of the encoder. Otherwise, the decoder asks the encoder to reveal the value of the *k*th bit in column 0. Sufficiently close is decided as follows:

(a) The decoder has a constant c, $c \leq 0.5$, a desired upper bound for the frame error rate, *U_FER*, and a gauge, unused_fer, the starting value of which is *U_FER*.
For $k = 0, 1, 2, \ldots, N - 1$,
(b) $x = \min(c, \text{unused\_fer}/(N - k))$;
(c) if ($p[k] > 1 - x$), slot k is assigned the value 0, and unused_fer = unused_fer − (1 – x);
(d) if ($p[k] < x$), slot k is assigned the value 1, and unused_fer = unused_fer – x;
(e) if ($x \leq p[k] \leq 1 - x$, the decoder demands that the encoder release the kth bit's value in column 0.

The linearity of the encoder implies that any plausible decoder executes clauses 1 and 2 M1 and M2 times, respectively, M1 + M2 < *N*. As shown in Sec. 4, if the constant δ is properly chosen, after the M1 instances of interactivity, Eve's knowledge will increase by up to M1 bits, while the decoder's knowledge will increase by at least M1-1 bits. The impact of the M2 instances of interactivity in column 0 cannot be estimated prior the I_D decoding. Nevertheless, the simulations run suggest that typically M2 ~ 0.12 × (M1 + M2). The impact of the M2 instances of interactivity on secrecy can be estimated during the decoding and guide the privacy amplification phase of the QKD protocol.

The simulations of the I_D algorithm suggest that

- the rules for column 0 are extremely conservative (in the batches run, the observed *FER* was, as a rule, smaller than *U_FER*/10), and
- the bulk of interactivity takes place out of column 0.

The following data are indicative of the typical outcome:
For $N = 2^{20}$, target *FER* = 0.04, and *QBER* = 0.04 after 1000 simulations:

- observed *FER* = 0;
- on average, there were 269,367 peeks to encoder's data (nearest integer), of which 33,664 (nearest integer) were in column 0; a perfect scheme would need peek at least 254,062 bit values,
- the observed γ value was 0.501; the maximum possible γ value is 0.515 (it should be noted that the maximum possible γ-values are limits as N tends to infinity.)

If the latency is not a factor, and the decoder can process the input values as fast as they are collected (e.g., by running multiple decoding threads in parallel), the I_D algorithm will outperform the limited-precision version. If not, one can decide which version of the polar decoder (interactive or classical and minimally interactive, as in Ref. [7]), best meets the needs of the QKD system in place.

In the following, the results collected for the interactive version and will show that even if is not retained for operational purposes, it can be used to design efficiently good sets of frozen bits.

3. Metrics of Performance

Metrics of performance were presented implicitly and explicitly in Ref. [6], the focus of which was to compare the low-density parity-check, LDPC, and the polar codes in the context of QKD reconciliation. Obviously, for any QKD implementation, one would like to minimize the latency and maximize the number of secret bits produced per unit of time. Given its focus, Ref. [6] proposed the following metrics:

1. $\beta = K/(N(1 - h(p)))$, where β shows how near the decoder is to the Shannon limit; and
2. the expected secrecy (measured in bits) of the decoder's output per unit of time.

The purpose here is more modest. Since the relevant metrics of an implementation depend on the environment, on the expected demands for keys, and on the hardware available (such as the ability to efficiently run multiple decoding threads in parallel), attention here is limited to the maximum number of secret bits one can produce. The metric, however, should allow direct comparison between algorithms using different *N* values. For this reason, the expected secrecy content per bit processed (γ in what follows) is used. As in Ref. [6], the signature_length value is dropped, because when $N \geq 2^{16}$, its impact is negligible. As a result, if a *K*/*N* encoding is used,

$$\gamma = (1 - FER)(K/N - h(p)),$$

and in the terms of the metrics in Ref. [4],

$$\gamma = (1 - FER)(\beta(1 - h(p)) - h(p)).$$

In what follows, γ as a metric of performance is used, but the *U_FER* value will also often be reported because the (*N*, *QBER*, *U_FER*) triplet guides the choice of the frozen bits and impacts the value of γ. The performance profiles reported in Ref. [6], augmented with the corresponding γ values, are as shown in Table 1.

Table 1. Performance profiles and corresponding γ values from Ref. [6].

N	*QBER*	β	*FER*	γ
2^{16}	0.02	0.935	0.09	0.602
2^{20}	0.02	0.963	0.11	0.610
2^{24}	0.02	0.98	0.08	0.644

Similar results to those of Ref. [6] were reported in Ref. [7], which used a different method for choosing the frozen bits and froze more bits (compare the β values) but obtained lower FER rates and marginally better yields

Table 2. Performance profiles and corresponding γ-values from Ref. [7].

N	*QBER*	β	*FER*	γ
2^{16}	0.02	0.93	0.073	0.609
2^{16}	0.04	0.901	0.071	0.409
2^{16}	0.06	0.878	0.068	0.290
2^{20}	0.02	0.962	0.086	0.626
2^{20}	0.04	0.945	0.092	0.430
2^{20}	0.06	0.931	0.092	0.271

4. The Impact of Interactivity in the I_D Algorithm

Wikipedia (at https://en.wikipedia.org/wiki/Binary_entropy_function) and Taylor's theorem, as well, inform us that the binary entropy function, $H_b(p) = -(p \times \log_2(p) + (1 - p) \times \log_2(1 - p))$, is in a neighborhood of 1/2 is equal to

$$H_b(p) = 1 - \frac{1}{2 \ln 2} \sum_{n=1}^{\infty} \frac{(1 - 2p)^{2n}}{n(2n - 1)},$$

and given that $(1/(2\ln(2))) < 1$ and that $n(2n - 1) \geq 1$, if $-\delta \leq 1 - 2p \leq \delta$, an $H_b(p) > 1 - \delta^2/(1 - \delta^2)$, it follows that if $M\,\delta^2/(1 - \delta^2) < 1$, and the decoder receives in the clear the values of *M* bits, the value of which is known by the decoder with probability *p* such that $(1 - \delta)/2 \leq p \leq (1 + \delta)/2$, then the eavesdropper will receive at most *M* bits of information, and the decoder will receive at least $(M - 1)$ bits of information.. The encoding of the polar codes is linear; as a result, any collection of *N* values in the encoding matrix either fully defines the contents of column 0 and of the matrix or are the values of entries that are linearly dependent. Therefore, any reasonable decoder will demand the values of *M*, where $M < N$, matrix values. As a result, if $\delta \leq (N - 1)^{-0.5}$, and *M* is as described above, the eavesdropper will receive at most *M* bits of information, and the decoder will receive at least $M - 1$.

5. Interactive Decoder's Performance

Simulations of the interactive version of the polar decoder produced the following indicative observed performance data and corresponding γ_max values (Table 3). The γ_max value, $1 - 2\times h(QBER)$, is the theoretical upper limit for γ as $N \rightarrow +\infty$ when a good set of frozen bits is used.

Table 3. Indicative observed performance data and corresponding γ_max values.

N, *p_rate*, *U_FER*, δ	Average peeks to column 0/Average of total peeks	f_rate	γ	γ_max
2^{16}, 0.02, 0.01, 2^{-8}	2 206.670/1085.496	0.000	0.697	0.717
2^{16}, 0.06, 0.01, 2^{-8}	4 274.076/23 667.681	0.001	0.311	0.345
2^{20}, 0.02, 0.01, 2^{-10}	21 776.258/159 517.176	0.000	0.706	0.717
2^{20}, 0.04, 0.01, 2^{-10}	34 677.084/270 380.156	0.000	0.500	0.515
2^{20}, 0.06, 0.01, 2^{-10}	44 613.777/363 029.560	0.000	0.326	0.345
2^{20}, 0.06, 0.20, 2^{-10}	44 613.777/360 421.119	0.000	0.329	0.345

The data in Table 3, collected through groups of 1000 simulations, suggest that good performance might be achieved if, at the very beginning of reconciliation, the decoder were given the values in the entries that are often the objects of interactive information exchanges. These entries could be anywhere in the sender's polar matrix and would be functionally the equivalent of the frozen bits/entries in the classical polar codes. Nevertheless, the raw data suggest that such a set of frozen bits may not exist. As an example, after 1000 simulations for the (2^{16}, 0.02, 0.01,2^{-8}) profile, the numbers of the observed interactivity instances within the ranges 0–0, 1–99, 100–199, 200–299, ..., 900–999, and 1000–1000 were {898 605, 125 860, 11 128, 5393, 2661, 17 282, 683, 199, 252, 323, 941, 749}.

It is worth noting, Table 4, that the interactive polar decoding works well for high *QBER* values, and the observed γ value increases as *N* increases.

Table 4. Performance of interactive decoding for high QBER values.

N	*QBER*	*U_FER*	Observed *FER*	γ	γ_max
2^{15}	0.08	0.01	0.000	0.1527	0.1956
2^{17}	0.08	0.01	0.000	0.1634	0.1956
2^{19}	0.08	0.01	0.000	0.1717	0.1956

It is also worth noting that the *U_FER* value is typically a very crude upper bound for the observed *FER*. As an example, for $N = 2^{16}$, $QBER = 0.02$, and $\delta = 2^{-8}$ 1000 simulations were run for each *U_FER* instance and there was a single failure to decode correctly, which was, for *U_FER* = 0.80.

Table 5. Impact of U_FER on the γ-value.

U_FER	Average calls to the encoder	Observed γ
0.02	10 524.735	0.6980
0.04	10 465.769	0.6989
0.08	10 403.272	0.6998
0.20	10 323.062	0.7010
0.40	10 260.286	0.7020
0.80	10 200.664	0.7022

6. Frozen Bit Sets Discovered Through Limited Precision and Interactivity

There are two reasons why the interactive approach requires less information and exhibits much better *FER* values. The first is that in columns 1 to $n - 1$, information is demanded when very little information for the bit's value is available. The second reason is the fact that the demands for information are tailored to the data received, while the frozen bits' approach must select a set of frozen bits that will work well with all possible error-patterns minus a set of error patterns of probability that does not exceed *U_FER*.

Nevertheless, the data suggest that some entries in the decoding matrix are more likely than others to be the subject of interactive demands for bit values. One may therefore ask the following questions:

- Can the interactive polar decoder provide information that enables the construction of good sets of frozen bits?
- Can the interactive polar decoder provide information that will enable us to improve the polar decoder by allowing us, in the QKD context, to mark as frozen entries anywhere in the encoding matrix?

The plausibility of these ideas was tested through decoding algorithm variants that proceeded as follows:

1. The entries of the decoding matrix are classified as **frozen** (values known prior to decoding), **known-value** (their value can be computed from the **frozen** bits), or **hidden-value**.
2. Simulations are run, and the interactive algorithm is called to retrieve **u** from the received vector, **v***. In each decoding instance,
   - o The bits in column 0 that demand interactivity are classified as **frozen**.
   - o At the end of each simulation, the bits for which values can be computed from the bits already classified as **frozen** are classified as **known-value**.
   - o For the **hidden-value** entries, the instances in which the decoder queried the value of the bits in question are counted.
3. Once sufficiently many simulations have been run, the **hidden-value** bits that have high counts (*e.g.*, 80 % of the simulations run in stage 2) are classified as **frozen**.
4. A new testing group of simulations is run. Their purpose is to mark as **frozen** the entries in column 0 that demand interactivity and to use the new **frozen** bits in order to expand the class of **known-value** bits. At this step, the limited-precision versions of the *f* and *g* transforms are used, and interactivity is used, as needed, only for column 0 entries.
5. Finally, a group of simulated encoding-transmission-decoding instances is run in order to estimate the *FER* value.

The simulations run showed that the entries that were promoted from **hidden-value** to **frozen** in step 3 fell in two categories:

a. If $M = 2^{(n-m)}$ is the smallest integer such that $(1-2p)^M < \delta$, the N/M entries 0, M, $2M$, , ..,, $N-M$ of column $n-m$ were marked **frozen** and remained **frozen.** Freezing these entries is equivalent to freezing the first N/M entries in column 0, i.e. the entries 0, 1, …, N/M -1 of column 0; a property that will be used later herein.

b. Barring the entries in (a) above, the few bits in columns $1-n$ that were initially marked **frozen** were eventually reclassified as **known-value** in step 4 above.

As a result, three closely connected algorithms were tested G0, G1, and G2, for the purpose of developing good sets of frozen entries:

G0: Bypasses steps 1–3 executes steps 4 and 5.
G1: Computes *M* as in (a), freezes the first *M* points in column 0, and runs steps 4 and 5.
G2: Computes *M*, freezes bits 0, *M*, 2*M*, 3*M*, …, $N-M$ in column $(n-m)$, executes steps 4 and 5.

Simulations showed that, as a rule, algorithms G0, G1, and G2 have similar performance levels and the performance data for algorithm G1 are used in the following. The following symbols are used:

- M_d, the numbers of simulations used to design a classical polar code for which the observed *FER* does not exceed *U_FER*.
- M_e, the number of simulations that will be run to estimate the code's *FER*.
- *K**, the number of the frozen bits, and
- *F*, the number of the observed decoding failures over *Me* decoding simulations.

The frozen bits in column #0 are determined through simulations as follows:

a. If side information is available (see, as an example, algorithm G1), the bits of known value in column 0 are marked as frozen.

b. The algorithm runs M_d instances of the P_D version of the classical polar decoder.

c. Variables unused_fer, *x*, and *p*[i] are defined and handled as in the I_D algorithm above; *i.e.*, for *i* = 0, 1,…,N − 1:
  (c1) if $p[i] < x$, unused_fer = unused_fer − $p[i]$ and the bit-value is set equal to 1;
  (c2) if $p[i]$ >(1 − x, unused_fer = unused_fer − $(1-p[i])$ and the bit-value is set equal to 0;. or

(c3) if $x \leq p[i] \leq 1 - x$, the *i*th bit of column 0 is marked as frozen and the correct bit value is inserted in the *i*th entry of column 0.

Table 6 shows the observed performance profiles (frozen bits, failures to decode correctly, γ) over nine combinations of (*U_FER*, M_d) while the other inputs (N, *QBER*, δ, M_e) are held fixed and equal to (2^{20}, 0.02, 2^{-10}, 640). We note that 640 simulations for $N = 2^{20}$ required ~15 minutes.

Table 6. Observed performance profiles.

U_FER	$M_d = 160$	$M_d = 320$	$M_d = 640$
0.01	189 749, 6, 0.6712	191 566, 4, 0.6716	193 262, 3, 0.6711
0.02	189 052, 8, 0.6698	190 843, 4, 0.6723	192 587, 4, 0.6707
0.04	188 370, 8, 0.6704	190 179, 4, 0.6730	19 1916, 4, 0.6713

These data (and those below) suggest that the search for perfection does not necessarily produce a better code.

In theory, QKD can create secrets, provided that $h(QBER) < 0.5$. Practice is another matter. If $QBER = 0.11$, γ_max ~ 0.00017, and $N = 2^{20}$, γ_max×N ~ 176.25. Practically no algorithm will be able to extract a meaningful secret unless it operates with higher N values. For $QBER = 0.10$, the maximum γ -value is $1 - 2 \times h(QBER) = 0.0620$. Table 7 suggests that for high *QBER*, the better profiles are found when the *U_ FER* parameter is high (even exceeding 1.0) and, as a result, the K/N ratio is, relatively speaking, high as well.

The simulation results cited below were obtained for $N = 2^{20}$, $\delta = 2^{-10}$, $QBER = 0.10$ (γ_max = 0.0620), and $M_e = 1024$.

Table 7. Observed performance profiles.

$N = 2^{20}$, $\delta = 2^{-10}$, $QBER = 0.10$	K^*, observed γ, errors per M_e simulations
U_FER = 0.25, $M_d = 512$	549 574, 0.0068, 14/1024
U_FER = 0.50, $M_d = 512$	548 673, 0.0076, 17/1024
U_FER = 0.75, $M_d = 512$	548 155, 0.0081, 19/1024
U_FER = 0.75, $M_d = 256$	545 795, 0.0101, 39/1024
U_FER = 1.00, $M_d = 512$	547 797, 0.0084, 20/1024

The simulation data suggest that for high *QBER*, the better profiles are found when the *U_FER* value is high and/or the design simulations are few. The actual data show that the coarsely derived sets of frozen bits perform better because they freeze fewer bits, while the strong *FER* proportional gains do not result in proportionally strong $(1 - FER)$ gains; e.g.,

1. (*U_FER* = 0.75, $M_d = 512$) will produce on average
   $(1 - 19/1024)(N - 548\,155 - N \times h(0.10)) \sim 8483.1$ bits.
2. (*U_FER* = 0.75, $M_d = 256$) will produce on average:
   $(1 - 39/1024)(N - 545\,795 - N \times h(0.10)) \sim 10\,584.4$ bits.

7. Appendix A: Basic Properties of the Polar Transform

If **u** is a vector of N entries, $N = 2^n$, each of which is equal to 0 or 1, and if

1. $\mathbf{u}^{\text{even}}$ and $\mathbf{u}^{\text{odd}}$ consist, respectively, of the $N/2$ even/odd indexed entries of **u**,
2. the symbol & represents vector concatenation, and
3. the symbol P_N is the operator that maps **u** onto **u**'s polar transform,

the N-point polar transform P_N is recursively defined as follows:

$$(\mathbf{u}_0, \mathbf{u}_1)P_2 = (\mathbf{u}_0 \oplus \mathbf{u}_1, \mathbf{u}_1) \tag{3a}$$

and

$$\mathbf{u}P_N = (\mathbf{u}^{even} \oplus \mathbf{u}^{odd})P_{N/2} \& u^{odd}P_{N/2} \qquad (3b)$$

Obviously, what is defined in Eq. (3) is a recursive transformation that can be implemented as a recursive routine. The tables below, 8 and 9, illustrate the Polar transform, and the "transformation of order 2" nature of the Polar transform; i.e. that for every bit-vector **u** of length N, $N=2^n$, $(\mathbf{u}P_N)P_N = \mathbf{u}$.

Table 8. Illustration of the Polar transform.

Col. 0	Col. 1	Col. 2	Col. 3
$\mathbf{u}_0$	$\mathbf{u}_0 \oplus \mathbf{u}_1$	$\mathbf{u}_0 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3$	$\mathbf{u}^*[0] = \mathbf{u}_0 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3 \oplus \mathbf{u}_4 \oplus \mathbf{u}_5 \oplus \mathbf{u}_6 \oplus \mathbf{u}_7$
$\mathbf{u}_1$	$\mathbf{u}_2 \oplus \mathbf{u}_3$	$\mathbf{u}_4 \oplus \mathbf{u}_5 \oplus \mathbf{u}_6 \oplus \mathbf{u}_7$	$\mathbf{u}^*[1] = \mathbf{u}_4 \oplus \mathbf{u}_5 \oplus \mathbf{u}_6 \oplus \mathbf{u}_7$
$\mathbf{u}_2$	$\mathbf{u}_4 \oplus \mathbf{u}_5$	$\mathbf{u}_2 \oplus \mathbf{u}_3$	$\mathbf{u}^*[2] = \mathbf{u}_2 \oplus \mathbf{u}_3 \oplus \mathbf{u}_6 \oplus \mathbf{u}_7$
$\mathbf{u}_3$	$\mathbf{u}_6 \oplus \mathbf{u}_7$	$\mathbf{u}_6 \oplus \mathbf{u}_7$	$\mathbf{u}^*[3] = \mathbf{u}_6 \oplus \mathbf{u}_7$
$\mathbf{u}_4$	$\mathbf{u}_1$	$\mathbf{u}_1 \oplus \mathbf{u}_3$	$\mathbf{u}^*[4] = \mathbf{u}_1 \oplus \mathbf{u}_3 \oplus \mathbf{u}_5 \oplus \mathbf{u}_7$
$\mathbf{u}_5$	$\mathbf{u}_3$	$\mathbf{u}_5 \oplus \mathbf{u}_7$	$\mathbf{u}^*[5] = \mathbf{u}_5 \oplus \mathbf{u}_7$
$\mathbf{u}_6$	$\mathbf{u}_5$	$\mathbf{u}_3$	$\mathbf{u}^*[6] = \mathbf{u}_3 \oplus \mathbf{u}_7$
$\mathbf{u}_7$	$\mathbf{u}_7$	$\mathbf{u}_7$	$\mathbf{u}^*[7] = \mathbf{u}_7$

* The change of color indicates different encoding segments within each column.

Table 9. Illustrating that the Polar transformation is of order 2.

Col. 0	Col. 1	Col. 2	Col. 3
u*[0] = **u** over {0,1,2,3, 4,5,6,7}	**u** over {0,1,2,3}	u over {0,1}	**u** over {0}
u*[1] = **u** over {4,5,6,7}	**u** over {2,3}	**u** over {1}	**u** over {1}
u*[2] = **u** over {2,3,6,7}	**u** over {1,3}	**u** over {2,3}	**u** over {2}
u*[3] = **u** over {6,7}	**u** over {3}	**u** over {3}	**u** over {3}
u*[4] = **u** over {1,3,5,7}	**u** over {4,5,6,7}	**u** over {4,5}	**u** over {4}
u*[5] = **u** over {5,7}	**u** over {6,7}	**u** over {5}	**u** over {5}
u*[6] = **u** over {3,7}	**u** over {5,7}	**u** over {6,7}	**u** over {6}
u*[7] = **u** over {7}	**u** over {7}	**u** over {7}	**u** over {7}

Nota bene: "**vector_name** over set S" stands for the "mode(2) sum of the **vector_name's** entries whose índices are in S; e.g., "**u** over {0,1,2,3, 4,5,6,7}" stands for $\mathbf{u}_0 \oplus \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3 \oplus \mathbf{u}_4 \oplus \mathbf{u}_5 \oplus \mathbf{u}_6 \oplus \mathbf{u}_7$ and "**u** over {5,7}" stands for **u**[5] ⊕ **u**[7]; moreover, we treat symbols such as $\mathbf{u}_{index}$ and **u**[index] as synonymous.

The reader can observe that the sets of indices in column 0 such those of Table 9, can be found as follows:

1. Form the binary expansion of the index.
2. Reverse the bits of the index.
3. Form all indices that can be obtained by flipping in all possible ways the 0 bits

Examples are:
0→000→000→xyz→{0,1,2,3,4,5,6,7}, 1→001→100→1yz→{4,5,6,7}
3→011→110→11z→{6,7} and 4→100→001→xy1→{1,3,5,7}
The same rule applies to the computations of **u*** from **u**; *i.e.*,
u*[0] is the parity of **u** over {0, 1, 2, … ,7}; **u***[1] is the parity of **u** over {4,5,6,7};
u*[3] is the parity of **u** over {6,7}; **u***[4] is the parity of **u** over {1,3,5,7}.

The entries of Table 8 are also the parities of u over subsets of {0,1,2,…,N-1}. For N=8, these subsets are shown in Table 9, beneath:

The following properties of the polar transform are significant; item 3, in particular, is the foundation of the polar decoder and the *f*/*g* probability computations:

1. The P_N transform is its own inverse; *i.e.*, if **u** is a bit-vector of *N* entries, and $\mathbf{u}P_N = \mathbf{u}^*$, then $\mathbf{u}^*P_N = \mathbf{u}$.

2. For $j = 0, 1, \ldots, n - 1$, the j-th column consists of 2^j consecutive segments, each of which contains $M = 2^{(n-j)}$ bit values.
3. If the bit values in column j are known, $0 \leq j < n$, the bit-values of column $(j + 1)$ are computed as follows:
   - If X and Y are entries $jM + 2t$, where $0 \leq t < M/2$, and $Y = X + 1$ in column j, and if X^* and Y^* are entries $jM + t$ and $X^* + M/2$ in column $j + 1$, then, by definition:
     - bit[X^*] = bit[X] $\oplus$ bit[Y] and bit[Y*] = bit[Y] (note that $1 \oplus 1 = 0$), and therefore
     - bit[X] = bit[X*] $\oplus$ bit[Y*] and bit[Y] = bit[Y*].
   - For any quadruple such as (X, Y, X^*, Y^*) above and S = {0,1, 2, 3,…,$N - 1$}, the following hold:

     There are subsets S[X*] and S[Y*] of S such that:
     - S[X*] and S[Y*] have the same cardinality;
     - the intersection of S[X*] and S[Y*] is the void set;
     - the value bit[X*] is the parity of **u*** over S[X*], and
     - the value bit[Y*] is the parity of **u*** over S[Y*].
     - the QKD polar decoder sees the values sent by the encoder as independent random variables;
     - in the absence of side information, the decoder treats bit[X*] and bit[Y*], the parities of non-overlapping sets, as independent random variables; and
     - the preceding properties are essential for the f and g computations of the polar decoder.

8. References

[1] Arıkan E (2008) Channel polarization: A method for constructing capacity-achieving code. *IEEE International Symposium on Information Theory (ISIT)*, pp 1173–1177. https://doi.org/10.1109/TIT.2009.2021379.

[2] Arıkan E, Telatar E (2009) On the rate of channel polarization. *IEEE International Symposium on Information Theory (ISIT)*, pp 1493–1495. https://doi.org/10.1109/ISIT.2009.5205856.

[3] Korada SB (2009) *Polar Codes for Channel and Source Coding*. Ph.D. dissertation. École Polytechnique fédérale *de* Lausanne, Lausanne, Switzerland. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.4335&rep=rep1&type=pdf.

[4] Hassani SH, Alishahi K, Urbanke RL (2014) Finite-Length Scaling for Polar Codes. *IEEE Transactions on Information Theory* 60(10):5875-5898. https://doi.org/10.1109/tit.2014.2341919.

[5] Hassani SH , Urbanke R (2012) Polar codes: Robustness of the successive cancellation decoder with respect to quantization. *2012 IEEE International Symposium on Information Theory Proceedings*, pp 1962-1966. https://doi.org/10.1109/ISIT.2012.6283642.

[6] Jouguet P, Kunz-Jacques S (2013) High performance error correction for quantum key distribution using polar codes. *ArXiv e-prints*. https://arxiv.org/abs/1204.5882v3.

[7] Nakassis A, Mink A (2014) Polar codes in a QKD environment. *Proceedings of SPIE: Defense Security & Sensing*, Baltimore, MD, Vol. 9123, pp 912305-1 to 912305-11.

[8] Bennett CH, Brassard G (1984) Quantum cryptography: Public key distribution and coin tossing. *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, Bangalore, India, p. 175.

[9] Bennett CH, Brassard G, Crepeau C, Maurer UM (1995) Generalized privacy amplification. *IEEE Transactions on Information Theory* 41(6):1915-1923. https://doi.org/10.1109/18.476316.

[10] Brassard G, Salvail G (1994) Secret-key reconciliation by public discussion. *Eurocrypt 1993*, Lofthus, Norway, pp 410–423.

[11] Gallager RG (1963) *Low-Density Parity-Check Codes*. Ph.D. thesis. Massachusetts Institute of Technology, Cambridge, MA. http://www.rle.mit.edu/rgallager/documents/ldpc.pdf.

[12] Nakassis A, Mink A (2012) LDPC error correction in the context of Quantum Key Distribution. *Proceedings of SPIE: Defense Security & Sensing*, Baltimore, MD.

[13] Richardson T, Shokrollahi M, Urbanke R (2001) Design of capacity approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory* 47(2):619–637. https://doi.org/10.1109/18.910578.

[14] Martinez-Mateo J, Pacher C, Peev M, Ciurana A, Martin V (2015) Demystifying the information reconciliation protocol cascade. *Quantum Information & Computation* 15(5–6):453–477. http://www.rintonpress.com/xxqic15/qic-15-56/0453-0477.pdf.

[15] Pedersen TB, Toyran M (2015) High performance information reconciliation for QKD with CASCADE. *Quantum Information and Computation* 15(5–6):419-434. http://www.rintonpress.com/xxqic15/qic-15-56/0419-0434.pdf.

About the author: *Anastase Nakassis is a Computer Scientist at the Advanced Network Technologies Division of the Information Technology Laboratory of NIST. The National Institute of Standards and Technology is an agency of the U.S. Department of Commerce.*