Open Source LDPC Error Correction for QKD

MAX-PLANCK-INSTITUT FÜR QUANTENOPTIK MPQ

Elitenetzwerk Bayern

LMU LUDWIG-MAXIMILIANS UNIVERSITÄT MÜNCHEN

dtec.bw Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr

Adomas Baliuka [2,3], **Elsa Dupraz** [1], **Rengaraj Govindaraj** [2,3], **Michael Auer** [2,3,5], **Peter Freiwang** [2,3], **Lukas Knips** [2,3,4] **and Harald Weinfurter** [2,3,4]

[1] IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France
[2] Ludwig-Maximilian-University (LMU), Munich, Germany
[3] Munich Center for Quantum Science and Technology (MCQST), Munich, Germany
[4] Max Planck Institute of Quantum Optics (MPQ), Garching, Germany
[5] Universität der Bundeswehr München, Neubiberg, Germany

IMT Atlantique Bretagne-Pays de la Loire École Mines-Télécom

xqp experimental quantum physics xqp.physik.uni-muenchen.de

MCQST

Abstract

Error correction is an essential step in the classical post-processing of all quantum key distribution (QKD) protocols. We present error correction methods optimized for discrete variable (DV) QKD and make them freely available as an ongoing open-source project (**github.com/XQP-Munich/LDPC4QKD**).

LDPC codes are the subject of active research with many applications, such as for Wi-Fi and digital television. They have been used for QKD error correction for a while, together with methods such as Cascade [2]. A single LDPC code operates on a fixed number of symbols and is optimized for a specific noise level of the quantum channel. In practice, the quality of the quantum channel fluctuates over time and across applications of a single QKD system. Rate adaption solves this issue by modifying a single LDPC code to adjust it to the current channel. We make use of recent, dedicated rate adaption methods specialized for Slepian-Wolf coding [1]. These offer advantages [3, 4] over most standard methods (e.g. puncturing and shortening) used in forward error correction and so far also for QKD error correction.

We invite contributions from the research community and plan to add support for more protocols, such as CV-QKD, in the future, incorporating further developments in QKD and channel coding.

Multi-Edge Type Protograph and Quasi-Cyclic LDPC Codes

QKD error correction using LDPC codes

- Suppose Alice and Bob each have a string of bits (sifted keys) of length N that are identical, except for the ratio of wrong bits in the string, the **quantum bit error ratio** (QBER).
- To reconcile the two, Alice sends Bob a sequence of bits (the syndrome) of length M, which is the matrix-vector product (mod 2) of her key with an $M \times N$ parity check matrix H. We call $R = \frac{M}{N}$ the **(leak) rate** of H.
- Bob uses a decoding algorithm (e.g. belief propagation) to correct his key to match Alice's syndrome. The probability of decoding failure is called the **frame error rate** (FER).
- Most decoding failures happen when the decoding algorithm fails to converge. Nevertheless, in a QKD protocol, eror correction using LDPC codes must be followed by a verification step.

LDPC code construction from protographs

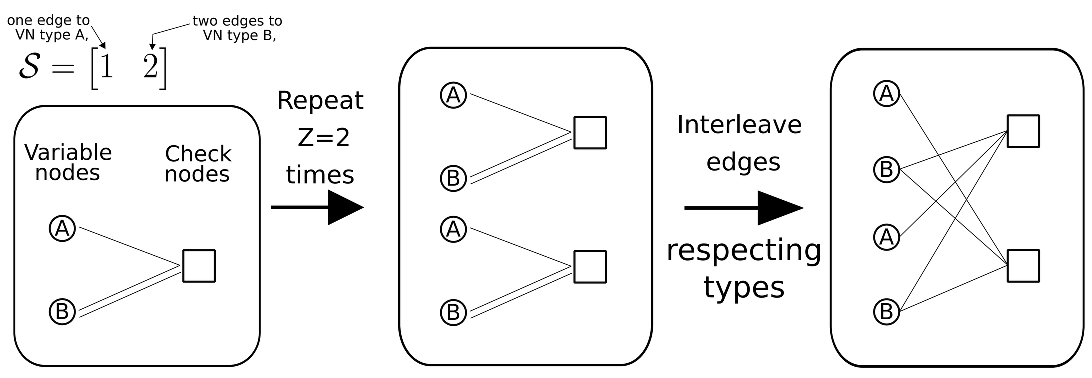
- A protograph [8] is a small matrix with integer coefficients that describes the degree distributions for a parity check matrix.
- Each row of the protograph represents a type of check node (CN); each column represents a type of variable node (VN).
- To construct an LDPC matrix, the protograph structure is repeated Z times and edges between nodes of corresponding types are interleaved.
- Interleaving is done via a progressive edge growth (PEG) algorithm. This allows the creation of a matrix with the correct degree distubution and few short cycles in the Tanner graph (important for good decoding performance using Belief Propagation).

Quasi-cyclic LDPC codes

- Quasi-cyclic LDPC codes [9] are a structured class of LDPC codes. Their parity check matrix is restricted to be a block matrix of circulant matrices.
- This structure allows memory-efficient storage of the matrix and efficient syndrome computation.
- It also allows lower complexity encoding when using a generator matrix, which is beneficial for forward error correction. In our application the generator matrix is not used.
- Our quasi-cyclic codes are lifted from the protograph LDPC codes (created as described above) using methods similar to [9].

Example construction from protograph

- Example (adapted from [1]): protograph $\mathcal{S} = [1 \;\; 2]$ specifies one type of CN and two types (called A, B) of VNs.



Protograph Optimization

Protograph creation

- Protographs with good thresholds constructed via a genetic algorithm (Differential Evolution [7]) and tested via Density Evolution.

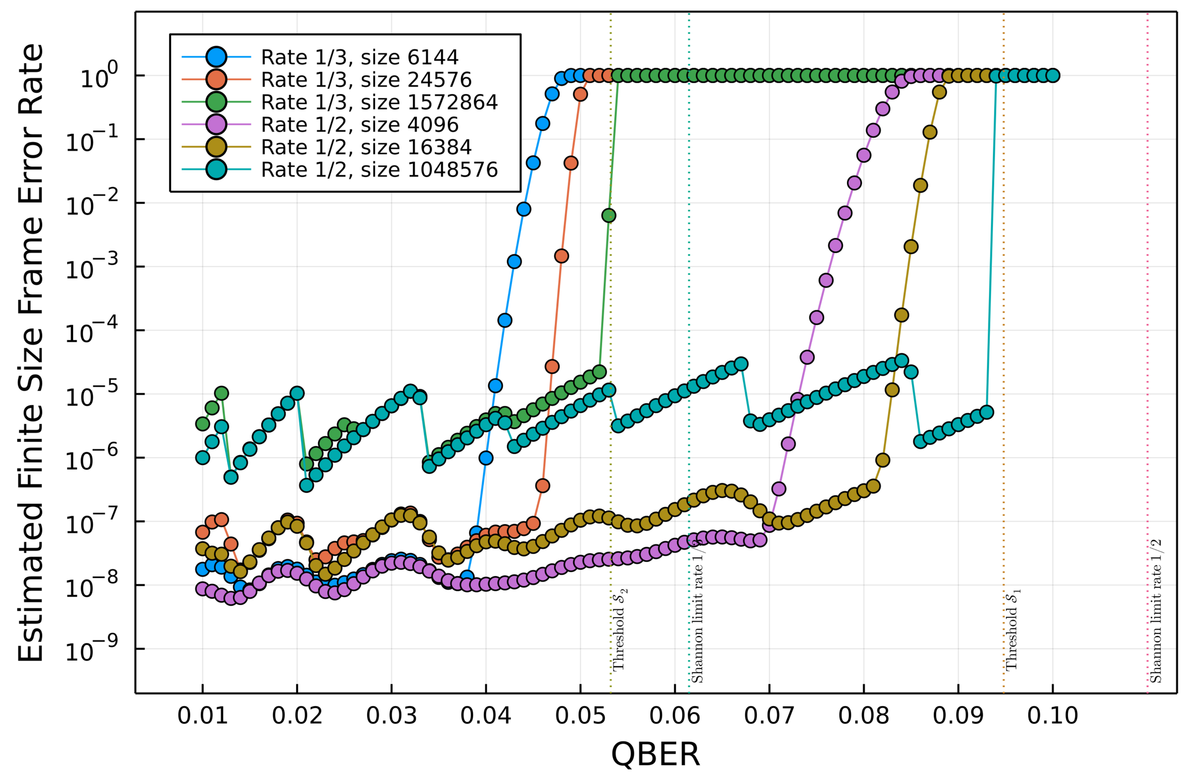
Protographs with rates 1/2 and 1/3

$$\mathcal{S}_1 = \begin{bmatrix} 2 & 3 & 2 & 4 \\ 1 & 0 & 2 & 5 \end{bmatrix} \qquad \mathcal{S}_2 = \begin{bmatrix} 3 & 1 & 3 & 4 & 2 & 2 \\ 4 & 1 & 0 & 4 & 0 & 1 \end{bmatrix}$$

- BSC thresholds (Density Evolution): **9.48**% for $\mathcal{S}_1$ (from [1]) and **5.32**% for $\mathcal{S}_2$.

Finite length performance

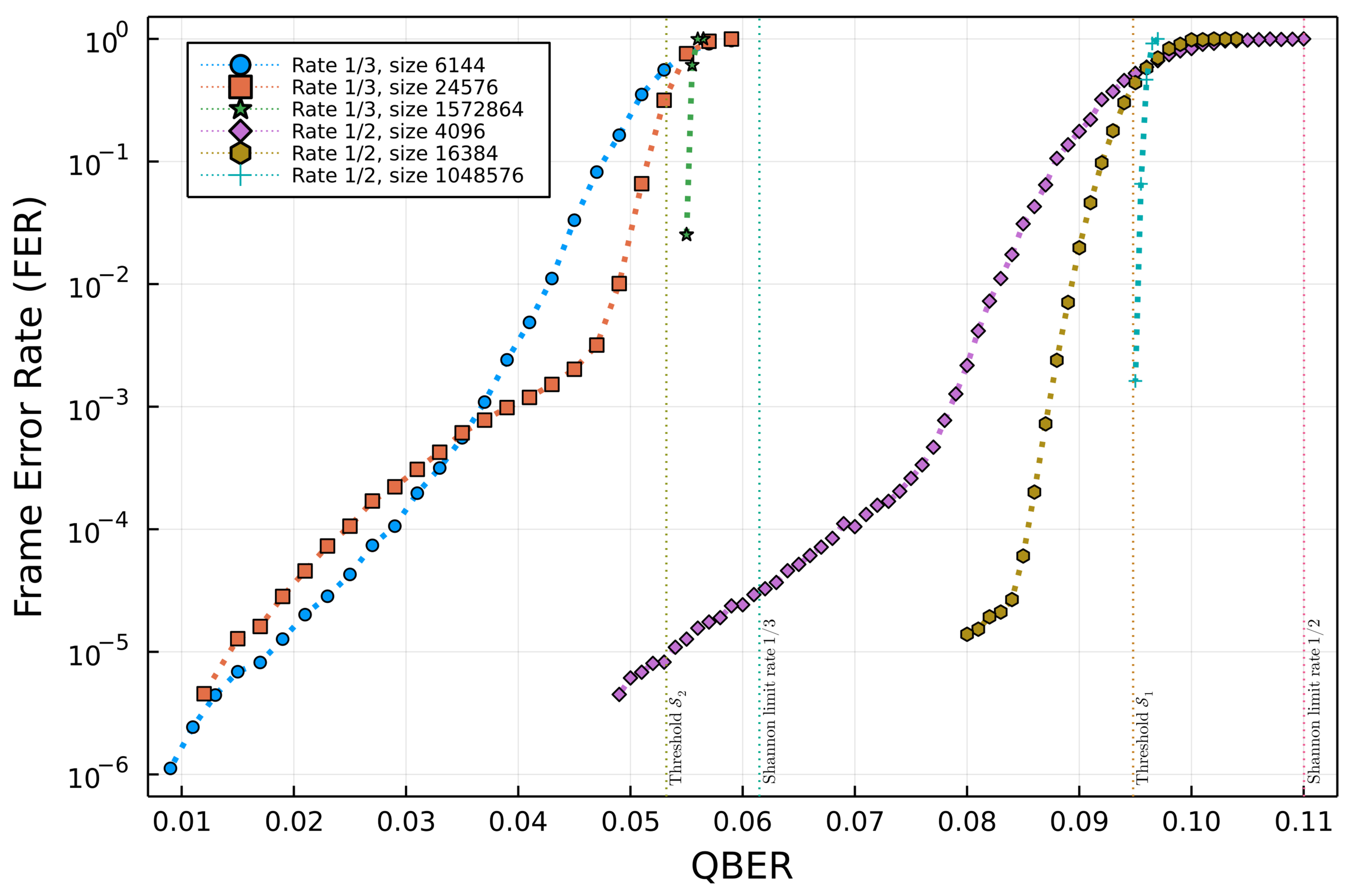
- Performance estimates [5] (using Density Evolution) for the block lengths considered in the construction.



Performance of constructed LDPC Codes

Decoding using belief propagation

- Frame error rates for different codes (varying rates and sizes) are compared.
- Simulations performed using `AFF3CT` [6] for better reproducibility (for each reported FER, at least 400/FER frames were simulated).
- Detailed simulation parameters and outputs are available in the repository.



Rate Adaption [1]

Need for rate adaption

- For error correction, a syndrome of the sifted key is exchanged. The syndrome length is given by the number of rows in the parity check matrix.
- For fixed QBER, too short syndromes lead to frame errors, while too long syndromes are inefficient by leaking more information to an eavesdropper than necessary.
  $\Rightarrow$ adapt syndrome length to current quantum channel

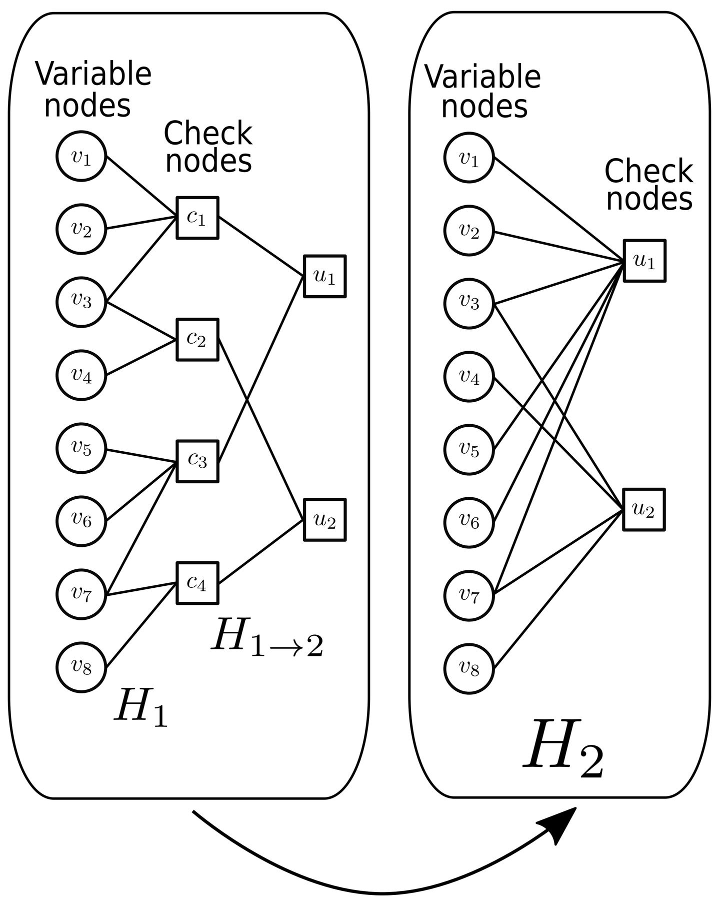
Rate adaptive code construction

- Given "mother" matrix H_1 with syndrome length m_1, obtain "daughter" matrix H_2 via an intermediate matrix $H_{1\to 2}$:

$$H_2 = H_{1\to 2} H_1$$

- If $H_{1\to 2}$ has size $m_2 \times m_1$, the rate adapted code H_2 uses smaller syndrome length m_2. This procedure is continued to obtain more different rates.
- The intermediate matrix $H_{1\to 2}$ should have full rank. This enables the receiver to uniquely recover the syndrome of H_1 from the syndrome of H_2, together with some additional syndrome bits from H_1.
- The Tanner graph of $H_{1\to 2}$ can be constructed from an intermediate protograph $\mathcal{S}_{1\to 2}$.
- We limit the possible $\mathcal{S}_{1\to 2}$ to have one or two values 1 in each row and zeros otherwise. With this, each rate adaption step amounts to combining two parity check equations of the mother matrix, selected from types given by the protograph and to minimize short cycles.

Combination of Tanner graphs

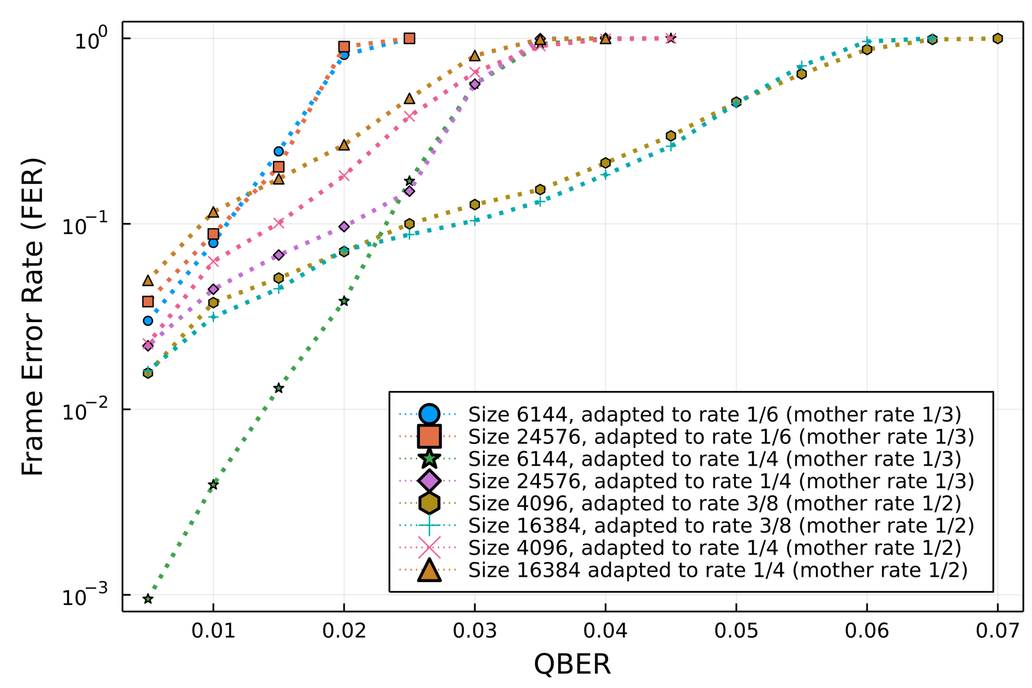


Rate Adapted Performance

Rate adapted codes

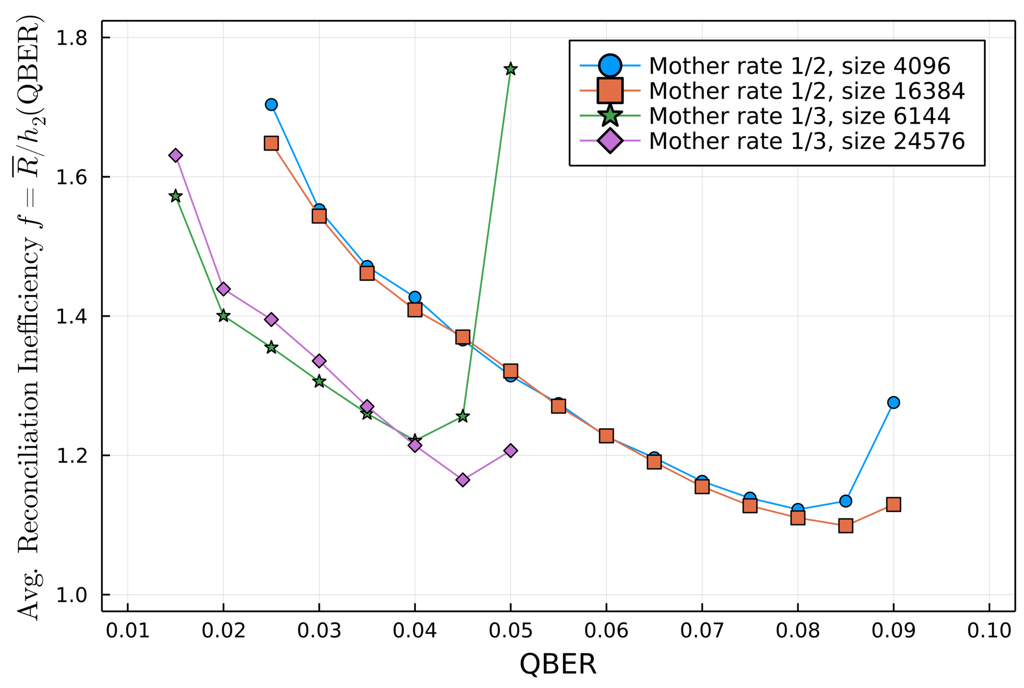
- We rate adapt each mother matrix to half its original rate (the rate adaption technique allows further rate reduction) in steps of one bit.
- Shown is rate adapted performance for the four smaller matrices. See the repository for more details.

Frame Error rates of rate adapted codes



Reconciliation inefficiency

- Let M be the syndrome length used to reconcile a key of length N.
- The **reconciliation inefficiency** is $f = \frac{M}{N\, h_2(\mathrm{QBER})} = \frac{R}{h_2(\mathrm{QBER})}$.
- Goal: as small as possible inefficiency f by minimizing rate $R = \frac{M}{N}$.
- Consider the **average leak rate** $\overline{R}$ under optimal amount of rate adaption, counting frame errors as $R = 1$ (similar to [2]).



References

[1] F. Ye, E. Dupraz, Z. Mheich, K. Amis, *IEEE Trans. Comm.* **67**, 3879 (2019).

[2] J. Martinez-Mateo, *et al.*, *Quantum Info. Comput.* **15**, 453 (2015).

[3] A. Liveris, Z. Xiong, C. Georghiades, *IEEE Comm. Letters* **6**, 440 (2002).

[4] D. Varodayan, A. Aaron, B. Girod, *Signal Processing* **86**, 3123 (2006).

[5] F. Leduc-Primeau and W. J. Gross, *ISTC* **9**, 325 (2016). doi: 10.1109/ISTC.2016.7593130

[6] A. Cassagne, *et al.*, *SoftwareX* **10**, 100345 (2019). doi: 10.1016/j.softx.2019.100345

[7] R. Storn and K. Price, *Proc. IEEE Evolutionary Computation* 842, (1996). doi: 10.1109/ICEC.1996.542711.

[8] J. Thorpe, *IPN progress report* **42**, 154 (2003).

[9] D. G. M. Mitchell, *et al.*, *IEEE Trans. Inform. Th.* **60**, 10 (2014).

Acknowledgements

We gratefully acknowledge the computational and data resources provided by the Leibniz Supercomputing Centre (www.lrz.de) and funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2111 – 390814868, from the EU project OpenQKD and the BMBF project QUBE. A.B. acknowledges support by Elitenetzwerk Bayern.