An Information Reconciliation Protocol for Secret-Key Agreement with Small Leakage

Christoph Pacher[1,†], Philipp Grabenweger[1] Jesus Martinez-Mateo[2] and Vicente Martin[2]

***Abstract*—We report on a variant of the so-called `Cascade` protocol that is well-known for its usage as information reconciliation protocol in quantum cryptography. A theoretical analysis of the optimal size of the parity check blocks is provided. We obtain a very small leakage which is for block sizes of 2^{16} typically only 2.5% above the Shannon limit, and notably, this holds for a QBER between 1% and 50%. For a QBER between 1% and 6% the leakage is only 2% above the Shannon limit. As comparison, the leakage of the original `Cascade` algorithm is 20% (40%) above the Shannon limit for a QBER of 10% (35%).**

I. Introduction

A. Information Reconciliation

Let X and Y be binary random variables (RV) belonging to two parties, named Alice and Bob, respectively. We assume that $\Pr[X=0]=\Pr[X=1]=\frac{1}{2}$ and that Y results from the transmission of X over a binary symmetric channel with crossover probability p, denoted by BSC(p). The parties have observed the vectors $\mathbf{x},\mathbf{y}\in\{0,1\}^n$ which are the outcomes of n independent and identically distributed (i.i.d.) instances of X and Y, respectively. Alice and Bob are allowed to exchange an arbitrary number of messages over a noiseless channel and perform local computations to reconcile Bob's vector. Finally, Bob shall have obtained an estimate $\hat{\mathbf{x}}$ of Alice's vector $\mathbf{x}$ such that the error probability $\epsilon:=\Pr\{\mathbf{x}\neq\hat{\mathbf{x}}\}$ is negligible [1].

This problem typically occurs in secret-key agreement over noisy channels, and particularly in quantum key distribution [1].

Note that our problem to reconcile $\mathbf{y}$ to $\mathbf{x}$ is identical to finding the error word $\mathbf{x}\oplus\mathbf{y}$, where the binary operator $\oplus$ denotes XOR (sum modulo 2). Thus to simplify the description of our analysis we assume that Alice has transmitted the all-zero vector $\mathbf{x}=\mathbf{0}$ and analyze how she can locate all 1's in Bob's vector through only the exchange of parities.

B. Efficiency Measures

In the following we introduce several measures of efficiency for an information reconciliation (IR) protocol over the BSC(p). As first measure, the efficiency can be defined as the percentage of additional information disclosed over the Shannon limit, i.e. the ratio of the number of transmitted bits m and the (asymptotic) minimum of the leakage, $nH(X|Y)=nh(p)$,

$$\eta_{IR}=\frac{m}{nh(p)}, \tag{1}$$

where $h(p)$ denotes the binary Shannon entropy function given by $h(p):=-p\log_2 p-(1-p)\log_2(1-p)$.

The reconciliation efficiency can be also defined as the ratio of the maximal length of the secret key (taking into account the leakage m) and the capacity of the BSC(p)

$$\beta_{IR}=\frac{1-m/n}{1-h(p)}. \tag{2}$$

Note that, in general $\eta_{IR}\geq 1$ and $\beta_{IR}\leq 1$, and the equality holds in both cases for perfect reconciliation.

II. Cascade as Linear Block Codes

In what follows we restate the `Cascade` protocol [1] in the language of linear block codes.

`Cascade` works in successive passes. At the beginning of pass i, Alice and Bob agree on the parity-check (PC) matrix $\mathbf{H}_i$ of a linear code with constant row weight k_i, such that each bit is covered by exactly one PC equation. Then Alice and Bob exchange the corresponding syndromes, $\mathbf{H}_i\mathbf{x}$ and $\mathbf{H}_i\mathbf{y}$, and for each different syndrome bit they perform a dichotomic search[1] (by exchanging further parities) to localize a bit error. After all errors positions are located Bob flips the corresponding bits in $\mathbf{y}$ such that the syndromes of Alice and Bob are now identical. They either stop the protocol or start a new pass with a new (random) PC matrix that covers different and larger sets of bits in each PC row.

In the original description of `Cascade` the row weights k_i of the PC matrices $\mathbf{H}_i$ were chosen as follows. Based on an estimate $\hat{p}$ of the channel parameter p the row weight in the first pass is $k_1=[0.73/\hat{p}]$, where $[\,]$ denotes the nearest integer. In following passes the row weight is doubled, $k_i=2k_{i-1}$.

From the second pass onward, each detected error can be used to correct further errors in other already completed passes. Let us suppose that an error is detected during the second pass. Since in the first pass all PCs are satisfied, it means that this bit error was in the first pass covered by an odd number of additional bit errors and thus it remains undetected

[1] CP, PG are with the Digital Safety & Security Department, AIT Austrian Institute of Technology GmbH, Donau-City-Straße 1, 1220 Vienna, Austria. [2] JM, VM are with Fac. Informática, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain. † E-mail: christoph.pacher@ait.ac.at

[1] They (i) split the corresponding PC equation in two non-overlapping PC equations (they split the check node of the corresponding Tanner graph), (ii) calculate the parity of $\mathbf{x}$ and $\mathbf{y}$ corresponding to the first new PC equation, and (iii) exchange both parities. If the parities differ they continue their splitting and parity exchange in the first PC equation. Otherwise, there is an odd number of errors in set of bits covered by the second PC equation and they continue their splitting and parity exchange there. The parties continue until they have located the exact position of a bit error in at most $\lceil\log_2 k_i\rceil$ steps.

by the PC equation. Consequently, there must be at least a second error in the bits covered by that PC equation and it can be now corrected by additional dichotomic searches. Since every bit participates in exactly one PC row in every pass, if an error is detected in pass i this uncovers errors in all passes $1, \ldots, i-1$. Note that dichotomic searches are shorter for earlier passes, thus the cascade process should start always from the first pass continuing until pass $i-1$. Further note that since (and as long as) the dichotomic searches corresponding to different PC rows are independent, in each pass, these can be processed and communicated in parallel.

Finally, the protocol concludes when four passes have been completed. These four passes are empirically enough to remove "all" discrepancies in a frame of 10^4 bits lengths [2].

III. On the Optimal Efficiency of Cascade

Recently we have reported a detailed numerical analysis and review of the Cascade protocol and its modifications [2]. It was observed that the optimal efficiency is achieved using powers-of-two row weights, i.e. $k_i = 2^{K_i}$. In combination with an optimization of K_i for p from 1% to 11% efficiencies $\eta_{IR} \approx 1.04 \ldots 1.06$ (corresponding to $\beta_{IR} \approx 0.997 \ldots 0.94$) could be achieved for a frame length of $n = 2^{16}$.

A deeper understanding of this fact was, however, not known. Below we state and motivate rules 1 and 2 that help us to understand this improvement. In addition we report on further rules (the most important one is rule 4) that bring about a further reduction of the leakage. A detailed information theoretic motivation for these rules is given in the attached manuscript.

Rule 1. *To reconcile with a number of transmitted bits close to the theoretical minimum of $nh(p)$, we shall transmit (mostly) bits that contain (almost) one bit of information.*

Equivalently, the conditional probability of (most) transmitted bits to be a one given the values of all previously transmitted bits must be (close to) $\frac{1}{2}$.

Let the message T be the RV formed by the concatenation of all transmitted bits $T_1, \ldots, T_m$. The length m of T is minimal if T is incompressible, i.e. $m - 1 < H(T) \leq m$.

Note that for the BSC(p), $H(T) \geq nh(p)$. Let us first assume that $H(T) = m$. Then, since $H(M) = H(X_1) + \sum_{k=2}^{m} H(X_k | X_1 \ldots X_{k-1})$, all (conditional) entropies on the right hand side (there are m of them) must be equal to one, and all conditional probabilities $\Pr[X_k = 1 | X_1 \ldots X_{k-1}] = \frac{1}{2}$. □

Rule 2. *To optimize the information transmitted during dichotomic searches the number of bits that are covered by parity-check rows should be a power of two.*

Let the term t-check denote a parity-check row of weight t. The parity of a binary vector $\mathbf{x}$ with respect to (w.r.t.) a parity-check row $\mathbf{h}$ is the inner product $\mathbf{h} \cdot \mathbf{x}$ (taken mod 2).

Let us assume we have a vector $\mathbf{x}$ that consists of i.i.d. bits x_i with $\Pr[x_i = 1] = p$, $0 < p < \frac{1}{2}$. Further assume we have a $2n$-check $\mathbf{h}$, which splits into two disjoint n-checks $\mathbf{h}'$ and $\mathbf{h}''$, i.e. $\mathbf{h} = \mathbf{h}' \oplus \mathbf{h}''$. Then, for all values of n if $\mathbf{x}$ has *odd parity* w.r.t. $\mathbf{h}$ ($\mathbf{h} \cdot \mathbf{x} = 1$) we get (using the symmetry in $\mathbf{h}'$ and $\mathbf{h}''$) that

$$\Pr[\mathbf{h}' \cdot \mathbf{x} = 1 \wedge \mathbf{h}'' \cdot \mathbf{x} = 0 | \mathbf{h} \cdot \mathbf{x} = 1] = \frac{1}{2},$$
$$\Pr[\mathbf{h}' \cdot \mathbf{x} = 0 \wedge \mathbf{h}'' \cdot \mathbf{x} = 1 | \mathbf{h} \cdot \mathbf{x} = 1] = \frac{1}{2}. \quad (3)$$

Consequently, the parity $\mathbf{h}' \cdot \mathbf{x}$ (or equivalently the parity $\mathbf{h}'' \cdot \mathbf{x}$) carries exactly $h\left(\frac{1}{2}\right) = 1$ bit of information about $\mathbf{x}$, which is the optimal case.

In the attachment we have shown that this is the only optimal case: (i) splitting $2n$-checks with even parity and (ii) splitting parity-checks with odd weight and arbitrary parity always leads to (conditional) probabilities for the resulting parities which are strictly different from $\frac{1}{2}$. Consequently, these parity bits carry less than 1 bit of information. Since splitting $2n$-checks with odd parities w.r.t. $\mathbf{x}$ into two n-checks is the only case where we gain exactly one bit of information if we want to continue this optimal splitting until we end up with two 1-checks, by induction we need to start with a 2^K-check (where K is a positive integer) with odd parity w.r.t. $\mathbf{x}$.

Rule 3. *In general, choose the PC weights as large as possible, but not larger than half the total block size.*

Our numerical simulation resulted in the following optimal check weights for the first pass:

$$k_1 = \begin{cases} \min(2^{[\log_2(1/p)]}, n/2) & \text{if } p \leq 0.25, \\ \min(\max(1, 2^{[\log_2(1/p)]-1}), n/2) & \text{if } p > 0.25. \end{cases} \quad (4)$$

Note, the $\min$ function limits the check weight to half the vector length, the $\max$ function avoids check weights smaller than one.

Rule 4. *In the second pass adapt parity-check weights and patterns to conditional bit error probabilities.*

Note that after the first bisection step of a 2^K-check with odd parity we obtain a 2^{K-1}-check with even parity and a 2^{K-1}-check with odd parity. This process continues by recursively dividing the check with odd parity until we finally divide a 2-check with odd parity.

Consequently, after the last step we have obtained from the original 2^K-check with odd parity a set of $2^{K'}$-checks for all $K' \in \{1, \ldots, K-1\}$ that all have even parity and two definitely known single bits which have the values 0 and 1 (or 1 and 0), respectively. In addition we have the set of 2^K-checks that had even parity in the first pass and had not been divided.

The conditional probability that a bit is an error bit in a t-check $\mathbf{h}$ (with i.i.d. bits) with even parity does strongly depend on the PC weight t. Therefore, when the standard Cascade selects bits randomly in the second pass it "forgets" that bits in PC of larger weight with even parity are (much) more likely wrong than bits in PC with smaller weight with even parity.

We will follow Rule 1 and transform the non-uniform distribution to a set of distributions that are each uniform but with different bit error probabilities: After the first pass we construct bit groups $\mathcal{B}_{K'}$: one group for each $K' \in \{1, \ldots, K\}$

that consists of all bits checked by all $2^{K'}$-checks (recall that after the first pass all PC have even parity), and another bit group that contains all single bits with values which are already definitely known. Then we choose the PC weight and create random PC patterns for each bit group $\mathcal{B}_{K'}$ separately.

Rule 5. *Perform the next dichotomic search in the check with odd parity with smallest weight.*

Rule 6. *Avoid to transmit redundant information.*

IV. Simulation Results

Simulations were performed to cover the range of the channel parameter p from 0.01 up to 0.5. Random bit frames were generated simulating independent Bernoulli processes with success probability p. Simulations have been comprehensively computed to accurately estimate the reconciliation efficiency, η_{IR} and β_{IR}, frame error rate, ϵ, and bit error rate, ϵ_b. To calculate those quantities we have simulated 10^5 frames for $n = 2^{10}$ and $n = 2^{14}$ and 10^4 frames for $n = 2^{16}$.

Fig. 1 shows the average reconciliation efficiency η_{IR} as defined in Eq. (1) of the original `Cascade` [1] (black), the optimized version proposed in [2] (green) and the modified protocol proposed here (blue). Two frame lengths of $n = 2^{14}$ bits (solid lines) and $n = 2^{16}$ bits (dashed lines) were considered. As shown in Fig. 1, the efficiency of the modified protocol proposed here improves the values reported in [2] for the region of interest in QKD (zoomed area), but even more interesting is that it does not degenerate for greater values of p.

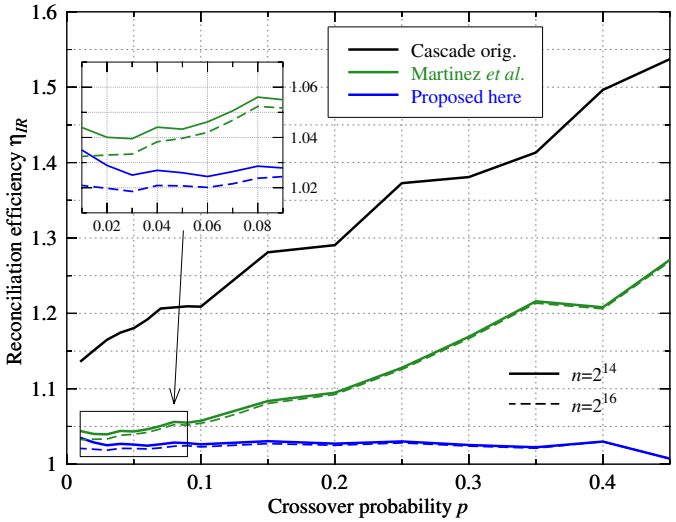

Fig. 1. Average reconciliation efficiency η_{IR}, Eq. (1), for the BSC(p).

Fig. 2 shows the average reconciliation efficiency as in Fig. 1 but considering the second definition for the efficiency given in Eq. (2). Note, that $\beta_{IR} > 0$ means that the leakage of the protocol is smaller than the channel capacity, and that a secret-key can be generated (neglecting other leakages).

Table I summarizes the efficiency, as defined in Eqs. (1) and (2), of the proposed reconciliation protocol together with the average block error probability for some remarkable crossover probabilities and several frame lengths.

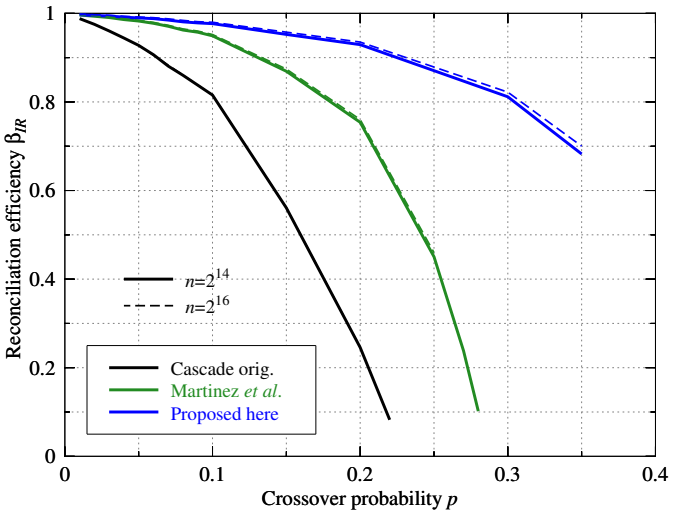

Fig. 2. Average reconciliation efficiency β_{IR}, Eq. (2), for the BSC(p).

TABLE I
Efficiency values, frame and bit error rates.

n	p	η_{IR}	β_{IR}	ϵ	ϵ_b
2^{10}	0.03	1.105	0.9747	1.6×10^{-4}	0.00146
2^{10}	0.1	1.064	0.9433	2.3×10^{-4}	0.00376
2^{10}	0.3	1.0466	0.9223	6×10^{-5}	0.00208
2^{14}	0.03	1.025	0.994	1.4×10^{-4}	0.0029
2^{14}	0.1	1.0263	0.9768	4×10^{-5}	0.00466
2^{14}	0.3	1.0254	0.8116	5×10^{-5}	0.0026
2^{16}	0.03	1.0185	0.9955	1×10^{-4}	0.0002
2^{16}	0.1	1.023	0.9798	0	0
2^{16}	0.3	1.024	0.822	0	0

A. Actual Leakage vs Number of Transmitted Bits

Given that we perform several random selections it may occur that a transmitted parity is linearly dependent on already transmitted parities. In such case, the actual information that the eavesdropper gets from this linear dependent bit is zero. Consequently, the leakage is bounded from above by the number of linear independent parities which is the rank over GF(2) of the binary matrix formed by all parity-check equations including also the dichotomic searches. Our simulations indicate that for values of the channel parameter satisfying $p < 0.1$ the quotient of the number of transmitted bits and the number of linearly independent bits is at the order of 1.001, and for larger values of p this quotient is smaller than 1.01, even for short frame lengths.

Acknowledgment

This work has been partially supported by the project Hybrid Quantum Networks, TEC2012-35673, funded by *Ministerio de Economía y Competitividad*, Spain.

References

[1] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *EUROCRYPT '93*, ser. Lecture Notes in Computer Science, vol. 765, 1994, pp. 410–423.

[2] J. Martinez-Mateo, C. Pacher, M. Peev, A. Ciurana, and V. Martin, "Demystifying the information reconciliation protocol Cascade," *Quantum Inform. Comput.*, vol. 15, no. 5&6, pp. 453–477, May 2015.

An Information Reconciliation Protocol for Secret-Key Agreement with Small Leakage

Christoph Pacher*‡, Philipp Grabenweger*, Jesus Martinez-Mateo† and Vicente Martin†
*Digital Safety & Security Department, AIT Austrian Institute of Technology GmbH,
Donau-City-Straße 1, 1220 Vienna, Austria.
†Facultad de Informática, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain.
‡Correspondence to: christoph.pacher@ait.ac.at

***Abstract*—We report on a highly efficient information reconciliation protocol for the binary symmetric channel (BSC) with feedback, proposed to be used in the context of secret-key agreement. This is a variant of the so-called `Cascade` protocol. Simulations determine efficiencies, defined by the ratio of actual transmitted information to the necessary amount of information, of approximately 1.025 for a frame length of 2^{14} bits and a frame error rate of typically 10^{-4}. The proposed algorithm works for any BSC parameter between 0 and 0.5.**

***Index Terms*—Information reconciliation, secret-key agreement, two-way reconciliation, cascade protocol, leakage.**

I. Introduction

We provide an algorithm and an analysis of it for the following information reconciliation (IR) problem:

Let X and Y be binary random variables (RV) belonging to two parties, named Alice and Bob, respectively. We assume that $\Pr[X=0]=\Pr[X=1]=\frac{1}{2}$ and that Y results from the transmission of X over a binary symmetric channel with crossover probability p, denoted by BSC(p). The parties have observed the vectors $\mathbf{x},\mathbf{y}\in\{0,1\}^n$ which are the outcomes of n independent and identically distributed (i.i.d.) instances of X and Y, respectively. Alice and Bob are allowed to exchange an arbitrary number of messages over a noiseless channel and perform local computations to reconcile Bob's vector. Finally, Bob shall have obtained an estimate $\hat{\mathbf{x}}$ of Alice's vector $\mathbf{x}$ such that the error probability $\epsilon:=\Pr\{\mathbf{x}\neq\hat{\mathbf{x}}\}$ is negligible [1].

This problem typically occurs in secret-key agreement over noisy channels, and particularly in quantum key distribution [1]. Therefore, in addition to having a low error probability ϵ, the amount of information disclosed about $\mathbf{x}$ that leaks on the noiseless channel, and the computational effort should be as small as possible.

A. Information Leakage

For any information reconciliation protocol in the context of secret-key agreement, a key figure is the amount of information that leaks to a potential eavesdropper during the execution of the protocol. Obviously, this number should be as small as possible, but often its exact determination is not possible and the number of transmitted bits is used as an upper bound. In some cases this estimate can, however, be improved. For example, in case that m linearly dependent parities are transmitted, the number m^* of *linearly independent* parities is clearly a sharper bound, since $m^*<m$ and linear combinations of bits do not contain additional information. Unfortunately, we are not aware of how to calculate this number during or after reconciliation without compromising the throughput of an IR protocol[1]. Therefore, we resort to using the number of transmitted bits in one direction[2] as an upper bound for the number of leaked bits and thus minimize the number of transmitted bits. As shown in Section VII-A, for most cases this upper bound is quite close to the actual number of linear independent bits.

B. General Notes

Note that our problem to reconcile $\mathbf{y}$ to $\mathbf{x}$ is identical to finding the error word $\mathbf{x}\oplus\mathbf{y}$, where the binary operator $\oplus$ denotes XOR (sum modulo 2). Thus to simplify the description of our analysis we assume that Alice has transmitted the all-zero vector $\mathbf{x}=\mathbf{0}$ and analyze how she can locate all 1's in Bob's vector through only the exchange of parities.

II. Theoretical Bounds

Recently, the IR problem in the finite block length case *without* feedback has been studied [4]–[6] and the following lower bound for the necessary amount of information m sent over the noise-free channel has been obtained [6]

$$\frac{m}{n}\geq H(X|Y)+\sqrt{\frac{V(X|Y)}{n}}\Phi^{-1}(1-\epsilon)-\frac{\log_2 n}{2n}-O\left(\frac{1}{n}\right). \tag{1}$$

where $\Phi(x)$ is the cumulative standard normal distribution,[3] H is the conditional entropy, and V is the channel dispersion. For discrete memoryless channels (DMC) under variable-length coding it is known that feedback does not increase the capacity C [7] nor the ϵ-capacity[4] [8], but that feedback in the finite block length case completely eliminates the term involving

[1]Gaussian elimination is one method to calculate the rank; asymptotically it requires the same number of algebraic operations as matrix multiplication [2].

[2]Note that, for an eavesdropper that has observed all messages on the noiseless channel from Alice to Bob, the messages that are transmitted from Bob to Alice provide no further information on Alice's vector $\mathbf{x}$. See also endnote [28] in [3].

[3]Often $Q(x):=1-\Phi(x)$ and the relation $\Phi^{-1}(1-\epsilon)=Q^{-1}(\epsilon)$ is used.

[4]The ϵ-capacity is defined as the capacity when an error probability ϵ is tolerated.

the *channel dispersion* [8]: for every DMC codes exist that use feedback and achieve block error rates ϵ with rates

$$R = \frac{C}{1-\epsilon} - O\left(\frac{\log n}{n}\right). \tag{2}$$

We are not aware of any theoretical result concerning bounds on m for finite-length IR with variable-length feedback, however it seems reasonable that a relation similar to Eq. (2), i.e. without the dispersion V will hold. The algorithm proposed here will indeed strengthen this conjecture.

III. Efficiency Measures

In the following we introduce several measures of efficiency for an information reconciliation protocol over the BSC(p). As first measure, the efficiency can be defined as the percentage of additional information disclosed over the Shannon limit, i.e. the ratio of the number of transmitted bits m and the (asymptotic) minimum of the leakage, $nH(X|Y) = nh(p)$,

$$\eta_{IR} = \frac{m}{nh(p)}, \tag{3}$$

where $h(p)$ denotes the binary Shannon entropy function given by $h(p) := -p\log_2 p - (1-p)\log_2(1-p)$.

The reconciliation efficiency can be also defined as the ratio of the maximal length of the secret key (taking into account the leakage m) and the capacity of the BSC(p)

$$\beta_{IR} = \frac{1-m/n}{1-h(p)}. \tag{4}$$

Note that, in general $\eta_{IR} \geq 1$ and $\beta_{IR} \leq 1$, and the equality holds in both cases for perfect reconciliation. Both measures η_{IR} and β_{IR} are related by

$$1 - \eta_{IR} h(p) = \beta_{IR}\left(1 - h(p)\right). \tag{5}$$

Further note that these measures are acceptable given that we are typically interested in very low error probabilities ϵ. However, as motivated by Eqs. (1) and (2), the error probability must be generally taken into account. In this regard, we additionally report here: (i) the block error probability ϵ, i.e. the ratio of frame pairs $(\mathbf{x}, \hat{\mathbf{x}})$ with $\mathbf{x} \neq \hat{\mathbf{x}}$ after reconciliation, and (ii) the bit error probability ϵ_b, i.e. the average ratio of different bits in both frames.

IV. The Cascade Protocol

We provide first a description of the original `Cascade` algorithm and some modified versions.

A. The Original Protocol

In what follows we restate the `Cascade` protocol [1] in the language of linear block codes.

`Cascade` works in successive passes. At the beginning of pass i, Alice and Bob agree on the parity-check (PC) matrix $\mathbf{H}_i$ of a linear code with constant row weight k_i, such that each bit is covered by exactly one PC equation. Then Alice and Bob exchange the corresponding syndromes, $\mathbf{H}_i\mathbf{x}$ and $\mathbf{H}_i\mathbf{y}$, and for each different syndrome bit they perform a dichotomic search[5] (by exchanging further parities) to localize a bit error. After all errors positions are located Bob flips the corresponding bits in $\mathbf{y}$ such that the syndromes of Alice and Bob are now identical. They either stop the protocol or start a new pass with a new (random) PC matrix that covers different and larger sets of bits in each PC row.

In the original description of `Cascade` the row weights k_i of the PC matrices $\mathbf{H}_i$ were chosen as follows. Based on an estimate $\hat{p}$ of the channel parameter p the row weight in the first pass is $k_1 = [0.73/\hat{p}]$, where $[\,]$ denotes the nearest integer. In following passes the row weight is doubled, $k_i = 2k_{i-1}$.

From the second pass onward, each detected error can be used to correct further errors in other already completed passes. Let us suppose that an error is detected during the second pass. Since in the first pass all PCs are satisfied, it means that this bit error was in the first pass covered by an odd number of additional bit errors and thus it remains undetected by the PC equation. Consequently, there must be at least a second error in the bits covered by that PC equation and it can be now corrected by additional dichotomic searches. Since every bit participates in exactly one PC row in every pass, if an error is detected in pass i this uncovers errors in all passes $1, \ldots, i-1$. Note that dichotomic searches are shorter for earlier passes, thus the cascade process should start always from the first pass continuing until pass $i-1$. Further note that since (and as long as) the dichotomic searches corresponding to different PC rows are independent, in each pass, these can be processed and communicated in parallel.

Finally, the protocol concludes when four passes have been completed. These four passes are empirically enough to remove "all" discrepancies in a frame of 10^4 bits lengths [3].

B. Earlier Improvements of the `Cascade` Protocol

Recently we have reported a detailed analysis and review of the `Cascade` protocol and its modifications [3]. Previous to [3] the main ideas for improving the protocol have been: (i) to optimize the row weights k_i and the number of passes, (ii) to replace the random construction of PC equations, (iii) to not include corrected bits in PC equations (that leads to a more accurate splitting into two PC equations covering the same number of potentially wrong bits) during the dichotomic searches, and (iv) to reuse PC equations where Alice and Bob have the same syndrome bit. In [3] a thorough numerical analysis based on simulations for the mentioned modifications has been provided. Focus was also put on accurately simulating the resulting block error probability ϵ of each variant. In addition, based on simulations it was observed in [3] that previous modifications of the row weights missed the

[5]They (i) split the corresponding PC equation in two non-overlapping PC equations (they split the check node of the corresponding Tanner graph), (ii) calculate the parity of $\mathbf{x}$ and $\mathbf{y}$ corresponding to the first new PC equation, and (iii) exchange both parities. If the parities differ they continue their splitting and parity exchange in the first PC equation. Otherwise, there is an odd number of errors in set of bits covered by the second PC equation and they continue their splitting and parity exchange there. The parties continue until they have located the exact position of a bit error in at most $\lceil \log_2 k_i \rceil$ steps.

most important point: the optimal efficiency is achieved using powers-of-two row weights, i.e. $k_i = 2^{K_i}$. In combination with an optimization of K_i for p from 1% to 11% efficiencies $\eta_{IR} \approx 1.04 \ldots 1.06$ (corresponding to $\beta_{IR} \approx 0.997 \ldots 0.94$) could be achieved for a frame length of $n = 2^{16}$.

V. On the Optimal Efficiency of Cascade

As commented above, recently we realized that the efficiency of `Cascade` significantly improves using parity-check weights that are powers of two [3]. A deeper understanding of this fact was, however, not known. Below we develop some ideas that help us understand this improvement.

Rule 1. *To reconcile with a number of transmitted bits close to the theoretical minimum of $nh(p)$, we shall transmit (mostly) bits that contain (almost) one bit of information.*

Equivalently, the conditional probability of (most) transmitted bits to be a one given the values of all previously transmitted bits must be (close to) $\frac{1}{2}$.

Let the message T be the RV formed by the concatenation of all transmitted bits $T_1, \ldots, T_m$. The length m of T is minimal if T is incompressible, i.e. $m - 1 < H(T) \leq m$.

Note that for the BSC(p), $H(T) \geq nh(p)$. Let us first assume that $H(T) = m$. Then, since $H(M) = H(X_1) + \sum_{k=2}^{m} H(X_k|X_1 \ldots X_{k-1})$, all (conditional) entropies on the right hand side (there are m of them) must be equal to one, and all conditional probabilities $\Pr[X_k = 1|X_1 \ldots X_{k-1}] = \frac{1}{2}$.

On the other hand if we transmit bits with conditional probabilities $q \neq \frac{1}{2}$, we transmit with each bit only $h(q)$ bits of information and we will need to transmit $\lceil nh(p)/h(q) \rceil$ bits.

A. Proof of Optimality of Dichotomic Searches on 2^K-Checks With Odd Parity

Let the term t-check denote a parity-check row of weight t. The parity of a binary vector $\mathbf{x}$ with respect to (w.r.t.) a parity-check row $\mathbf{h}$ is the inner product $\mathbf{h} \cdot \mathbf{x}$ (taken mod 2).

Let us assume we have a vector $\mathbf{x}$ that consists of i.i.d. bits x_i with $\Pr[x_i = 1] = p$, $0 < p < \frac{1}{2}$. Further assume we have a $2n$-check $\mathbf{h}$, which splits into two disjoint n-checks $\mathbf{h}'$ and $\mathbf{h}''$, i.e. $\mathbf{h} = \mathbf{h}' \oplus \mathbf{h}''$. Then, for all values of n if $\mathbf{x}$ has *odd parity* w.r.t. $\mathbf{h}$ ($\mathbf{h} \cdot \mathbf{x} = 1$) we get (using the symmetry in $\mathbf{h}'$ and $\mathbf{h}''$) that

$$\begin{aligned} \Pr[\mathbf{h}' \cdot \mathbf{x} = 1 \wedge \mathbf{h}'' \cdot \mathbf{x} = 0 | \mathbf{h} \cdot \mathbf{x} = 1] &= \frac{1}{2}, \\ \Pr[\mathbf{h}' \cdot \mathbf{x} = 0 \wedge \mathbf{h}'' \cdot \mathbf{x} = 1 | \mathbf{h} \cdot \mathbf{x} = 1] &= \frac{1}{2}. \end{aligned} \tag{6}$$

Consequently, the parity $\mathbf{h}' \cdot \mathbf{x}$ (or equivalently the parity $\mathbf{h}'' \cdot \mathbf{x}$) carries exactly $h\left(\frac{1}{2}\right) = 1$ bit of information about $\mathbf{x}$, which is the optimal case.

If $\mathbf{x}$ has *even parity* w.r.t. $\mathbf{h}$ ($\mathbf{h} \cdot \mathbf{x} = 0$) we have

$$\begin{aligned} \Pr[\mathbf{h}' \cdot \mathbf{x} = 0 \wedge \mathbf{h}'' \cdot \mathbf{x} = 0 | \mathbf{h} \cdot \mathbf{x} = 0] &= \frac{p_{\text{even}}^2(n,p)}{p_{\text{even}}(2n,p)} > \frac{1}{2}, \\ \Pr[\mathbf{h}' \cdot \mathbf{x} = 1 \wedge \mathbf{h}'' \cdot \mathbf{x} = 1 | \mathbf{h} \cdot \mathbf{x} = 0] &= \frac{p_{\text{odd}}^2(n,p)}{p_{\text{even}}(2n,p)} < \frac{1}{2}. \end{aligned} \tag{7}$$

Since both cases have probabilities different from one half the information contained in $\mathbf{h}' \cdot \mathbf{x}$ (or $\mathbf{h}'' \cdot \mathbf{x}$) is always smaller than one bit. Here

$$p_{\text{even}}(n,p) := \Pr[\mathbf{h} \cdot \mathbf{x} = 0] = \frac{1 + (1-2p)^n}{2}, \text{and} \tag{8}$$

$$p_{\text{odd}}(n,p) := \Pr[\mathbf{h} \cdot \mathbf{x} = 1] = \frac{1 - (1-2p)^n}{2} \tag{9}$$

denote the probabilities that $\mathbf{x}$ has even or odd parity w.r.t. an n-check $\mathbf{h}$, respectively.

Similarly, we can proof that any splitting of $(2n+1)$-checks with odd or even parity leads for $p \neq \frac{1}{2}$ always to conditional probabilities different from one half and thus results in parities with less than one bit of information.

So far we have seen that splitting $2n$-checks with odd parities w.r.t. $\mathbf{x}$ into two n-checks is the only case where we gain exactly one bit of information (cf. Eq. (6)). If we want to continue this optimal splitting until we end up with two 1-checks, by induction we need to start with a 2^K-check (where K is a positive integer) with odd parity w.r.t. $\mathbf{x}$. □

VI. Improved Protocol

As we already reported in [3], starting with a frame of 2^{14} bits and using only 2^K-checks could significantly improve the efficiency η_{IR} of `Cascade` (typically from 1.15 to 1.05). However, since the efficiency is still larger than unity, there must be further ways to improve this. In the following we give first an overview of where `Cascade` still "leaks" too much information, and second we describe how to fix it.

A. Optimize Initial Check Weights

Previously, all reported implementations of `Cascade` have used for the first pass k_1-checks where $k_1 < 2/p$, and most of them used $k_1 \leq 1/p$. However, note that $p_{\text{even}}(n,p)$ and $p_{\text{odd}}(n,p)$, cf. Eqs. (8) and (9), are both approaching $\frac{1}{2}$ with increasing n. Thus, in general, larger PC weights in the first round should be better (cf. Rule 1). However, this rule interferes with the rule of the next section: making PC weights too large reduces the advantage of the next step. In contrast our numerical simulation resulted in the following optimal check weights for the first pass:

$$k_1 = \begin{cases} \min(2^{[\log_2(1/p)]}, n/2) & \text{if } p \leq 0.25, \\ \min(\max(1, 2^{[\log_2(1/p)]-1}), n/2) & \text{if } p > 0.25. \end{cases} \tag{10}$$

Note, the $\min$ function limits the check weight to half the vector length, the $\max$ function avoids check weights smaller than one. Note also, that for $p > 0.355$ the optimal check weight is one, i.e. the complete vector is transmitted on the noiseless channel.

B. Group Bits According to Conditional Probability

We will have a closer look at the iterative bisection of 2^K-checks with odd parity w.r.t $\mathbf{x}$. Note that after the first bisection step we obtain a 2^{K-1}-check with even parity and a 2^{K-1}-check with odd parity. The 2^{K-1}-check with odd parity will itself be further divided into a 2^{K-2}-check with even parity

and a 2^{K-2}-check with odd parity. This process continues by dividing the check with odd parity until we finally divide a 2-check with odd parity.

Consequently, after the last step we have obtained from the original 2^K-check with odd parity a set of $2^{K'}$-checks (for all $K' \in \{1, \ldots, K-1\}$ we obtain exactly one PC) that all have even parity and two definitely known single bits which have the values 0 and 1 (or 1 and 0), respectively. In addition we have the set of 2^K-checks that had even parity in the first pass and had not been divided.

Now, after the first pass the standard `Cascade` algorithm continues by creating PC rows with larger weight covering randomly selected bits, calculating parities, dividing PC with odd parities, and performing dichotomic searches to localize further errors. However, already the first step (selecting bit positions randomly) is definitely non-optimal as we will show in the following.

Note, that the conditional probability that a bit in a t-check (with i.i.d. bits) with even parity is an error bit does strongly depend on the PC weight t

$$p_{\text{bit}}(t, p|\mathbf{h} \cdot \mathbf{x} = 0) = p\frac{p_{\text{odd}}(t-1, p)}{p_{\text{even}}(t, p)}. \tag{11}$$

Therefore, when selecting bits randomly for each new pass we "forget" that we have already learned in previous passes, i.e. that bits in PC of larger weight with even parity are (much) more likely wrong than bits in PC with smaller weight with even parity. In other words we destroy information.

To be able to transmit bits that have (almost) one bit of information (cf. Eq. (6)), we transform the non-uniform distribution to a set of distributions that are each uniform but with different bit error probabilities. After the first pass we construct bit groups $\mathcal{B}_{\mathcal{K}'}$: one group for each $K' \in \{1, \ldots, K\}$ that consists of all bits checked by all $2^{K'}$-checks (recall that after the first pass all PC have even parity), and another bit group that contains all single bits with values which are already definitely known. We have now obtained K different bit groups that can contain errors, each one consisting of bits with the same *a-posteriori* error probability that we can calculate according to Eq. (11).

In the second pass we adapt the PC weight $k_{2,K'}$ for each bit group $\mathcal{B}_{\mathcal{K}'}$ as

$$k_{2,K'} = \min(2^{[\log_2(4/p_{\text{bit}}(2^{K'}, p|\mathbf{h}\cdot\mathbf{x}=0))]}, |\mathcal{B}_{\mathcal{K}'}|/2), \tag{12}$$

and create random PC patterns for each bit group separately.

C. Perform Bisection on the Smallest of All Possible Checks

Let us assume that in i-th pass the dichotomic search has located a bit error. This error can be tracked back to exactly one PC equation per previous pass, i.e. by flipping the error bit it produces $i-1$ PC with odd parity. Potentially, we have found $i-1$ additional bit errors. However, note that it is possible that two or more PC equations contain the *same* bit error given that these PC are all from different passes.

In any case we have to decide how we should continue with the processing. If we want to minimize the number of transmitted bits it is necessary to bisect the smallest of all PC with odd parity, which could belong to any of the passes, from 1 to $i-1$, let us say pass i'. Bisecting this PC results in another localized bit error. Again this error is corrected in all other $i-1$ passes. We potentially obtain $i-1$ new bit errors and their corresponding PC with odd parity. However, as the process continues, it may also happen that PC with odd parity are affected and toggled back to even parity.

Let us for a moment concentrate on the simplest case, pass 2. If we localize an error in a PC of pass 2 we can correct the error in some $2^{K'}$-check of pass 1. This PC now has odd parity. Note that, one bit out of $2^{K'}$ bits is already exactly known before the dichotomic search. Nevertheless, we start a standard dichotomic search, however, we take care not to transmit redundant information. (see Section VI-D).

D. Reducing Redundant Information

Another improvement above the standard (state-of-the-art) `Cascade` consists in keeping a list of all frame bits which are already known to be correct. Whenever we correct an erroneous bit (which has been located by a previous bisection), we know that the bit is now definitely correct (it cannot happen in the `Cascade` algorithm that an already correct bit is toggled), thus we can add it to the list of known correct bits. Additionally, whenever we do a bisection of an odd PC, we learn the parity sums of the two halves of the PC. If the half with parity 0 consists only of a single bit, we can also add this bit to the list of correct bits. We can use this list of correct bits in the following manner: Whenever we do a bisection of an odd PC, we would in principle have to calculate the parity sum of one half of the PC (the parity sum of the other half is then also known, since the whole PC has odd parity). Before we do this, we can check if one of the halves consists only of bits which are already in the list of known correct bits. If this is the case, we know the parities of both halves and do not need to transmit any parity bit and can immediately proceed with bisecting the half with odd parity. The same can be done at the beginning of the i-th pass, when we start with calculating the parity sums over PC of size k_i. For each PC we can check if it consists only of already known correct bits, and if it does we do not need to transmit it.

If we use bit groups (cf. Section VI-B) in the second pass, we know that the parity sum of each bit group is even. Thus after new PCs are formed the parity of the last PC in each group is redundant and need not be transmitted.

VII. Simulation Results

Simulations were performed to cover the range of the channel parameter p from 0.01 up to 0.5. This is in contrast to previous studies that typically concentrate only on $p \in [0, 0.11]$, the interval mainly of interest for discrete variable QKD protocols, such as the well-known BB84 protocol [9]. Random bit frames were generated simulating independent Bernoulli processes with success probability p. Simulations have been comprehensively computed to accurately estimate the reconciliation efficiency, η_{IR} and β_{IR}, frame error rate,

ϵ, and bit error rate, ϵ_b. To calculate those quantities we have simulated 10^5 frames for $n = 2^{10}$ and $n = 2^{14}$ and 10^4 frames for $n = 2^{16}$.

Fig. 1 shows the average reconciliation efficiency of the original `Cascade` [1] (black), the optimized version proposed in [3] (green) and the modified protocol proposed here (blue). Two frame lengths of $n = 2^{14}$ bits (solid lines) and $n = 2^{16}$ bits (dashed lines) were considered. As shown in Fig. 1, the efficiency of the modified protocol proposed here improves the values reported in [3] for the region of interest in QKD (zoomed area), but even more interesting is that it does not degenerate for greater values of p.

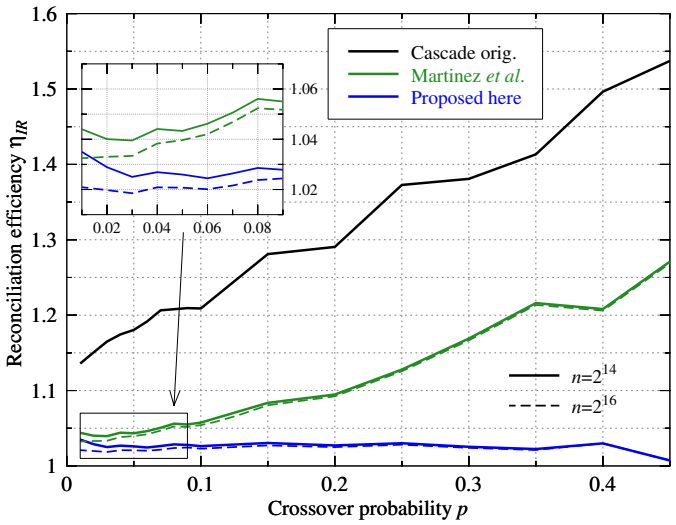

Fig. 1. Average reconciliation efficiency for the BSC(p).

Table I summarizes the efficiency, as defined in Eqs. (3) and (4), of the proposed reconciliation protocol together with the average block error probability for some remarkable crossover probabilities and several frame lengths.

TABLE I
EFFICIENCY VALUES, FRAME AND BIT ERROR RATES.

n	p	η_{IR}	β_{IR}	ϵ	ϵ_b
2^{10}	0.03	1.105	0.9747	1.6×10^{-4}	0.00146
2^{10}	0.1	1.064	0.9433	2.3×10^{-4}	0.00376
2^{10}	0.3	1.0466	0.9223	6×10^{-5}	0.00208
2^{14}	0.03	1.025	0.994	1.4×10^{-4}	0.0029
2^{14}	0.1	1.0263	0.9768	4×10^{-5}	0.00466
2^{14}	0.3	1.0254	0.8116	5×10^{-5}	0.0026
2^{16}	0.03	1.0185	0.9955	1×10^{-4}	0.0002
2^{16}	0.1	1.023	0.9798	0	0
2^{16}	0.3	1.024	0.822	0	0

A. Actual Leakage vs Number of Transmitted Bits

Given that we perform several random selections it may occur that a transmitted parity is linearly dependent on already transmitted parities. In such case, the actual information that the eavesdropper gets from this linear dependent bit is zero. Consequently, the leakage is bounded from above by the number of linear independent parities which is the rank over GF(2) of the binary matrix formed by all parity-check equations including also the dichotomic searches. Our simulations indicate that for values of the channel parameter satisfying $p < 0.1$ the quotient of the number of transmitted bits and the number of linearly independent bits is at the order of 1.001, and for larger values of p this quotient is smaller than 1.01, even for short frame lengths.

VIII. Conclusions

We have proposed and simulated an improved information reconciliation protocol with feedback that is well suited for the secret-key agreement problem. A previous observation on the optimality of using powers of two for the weights of the parity-checks in the `Cascade` protocol was motivated here for the first time. In additi n, our analysis paved the way for a set of additional improvements that helped to increase the efficiency of `Cascade`. For a frame length of 2^{14} the proposed protocol notably achieves an efficiency below $\eta_{IR} < 1.03$ in the range $p \in [0.02, 0.5]$. This closed approximately half of the previously remaining gap to an efficiency of unity at the beginning of the range, but also remarkably, the efficiency remains constant for higher values of p.

Acknowledgment

This work has been partially supported by the project Hybrid Quantum Networks, TEC2012-35673, funded by *Ministerio de Economía y Competitividad*, Spain.

References

[1] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *EUROCRYPT '93*, ser. Lecture Notes in Computer Science, vol. 765, 1994, pp. 410–423.

[2] J. R. Bunch and J. E. Hopcroft, "Triangular factorization and inversion by fast matrix multiplication," *Math. Comp.*, vol. 28, pp. 231–236, 1974.

[3] J. Martinez-Mateo, C. Pacher, M. Peev, A. Ciurana, and V. Martin, "Demystifying the information reconciliation protocol Cascade," *Quantum Inform. Comput.*, vol. 15, no. 5&6, pp. 453–477, May 2015.

[4] M. Hayashi, "Second-order asymptotics in fixed-length source coding and intrinsic randomness," *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4619–4637, Oct. 2008.

[5] V. Tan and O. Kosut, "The dispersion of Slepian-Wolf coding," in *IEEE Int. Symp. Inf. Theory*, July 2012, pp. 915–919.

[6] M. Tomamichel, J. Martinez-Mateo, C. Pacher, and D. Elkouss, "Fundamental finite key limits for information reconciliation in quantum key distribution," in *IEEE Int. Symp. Inf. Theory*, June-July 2014, pp. 1469–1473.

[7] C. E. Shannon, "The zero error capacity of a noisy channel," *IRE Trans. Inf. Theory*, vol. 2, no. 3, pp. 8–19, 1956.

[8] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Feedback in the non-asymptotic regime," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 4903–4925, Aug. 2011.

[9] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *IEEE Int. Conf. Computers, Systems, & Signal Processing*, Dec. 1984, pp. 175–179.