This is the accepted manuscript made available via CHORUS. The article has been published as:

Family of finite geometry low-density parity-check codes for quantum key expansion

Kung-Chuan Hsu and Todd A. Brun

Phys. Rev. A **87**, 062332 — Published 24 June 2013

DOI: 10.1103/PhysRevA.87.062332

Family of Finite Geometry Low-Density Parity-Check Codes for Quantum Key Expansion

Kung-Chuan Hsu* and Todd A. Brun†
Ming Hsieh Department of Electrical Engineering,
University of Southern California, Los Angeles, California 90089, USA

We consider the quantum key expansion (QKE) protocol based on entanglement-assisted quantum error-correcting codes (EAQECCs). In these protocols, a seed of previously shared secret key is used in the post-processing stage of a standard quantum key distribution protocol like BB84, in order to produce a larger secret key. This protocol was proposed by Luo and Devetak, but codes leading to good performance have not been investigated. We look into a family of EAQECCs generated by classical finite geometry (FG) low-density parity-check (LDPC) codes, for which very efficient iterative decoders exist. A critical observation is that almost all errors in the produced secret key result from uncorrectable block errors that can be detected by an additional syndrome check and an additional sampling step. Bad blocks can then be discarded. We make some changes to the original protocol to avoid the consumption of secret key when the protocol fails. This allows us to greatly reduce the bit error rate of the key at the cost of a minor reduction in the key production rate, but without increasing the consumption rate of pre-shared key. We present numerical simulations for the family of FG LDPC codes, and show that this improved QKE protocol has a good net key production rate even at relatively high error rates, for appropriate choices of these codes.

PACS numbers: 03.67.Dd,03.67.Hk,03.67.Ac,03.67.Pp

I. INTRODUCTION

A quantum key expansion protocol allows two parties, Alice and Bob, to expand a shared secret key by using one-way quantum communication and public classical communication. Luo and Devetak [1] demonstrated a QKE protocol, which is derived from the standard BB84 quantum key distribution (QKD) protocol with post-processing steps involving the use of entanglement-assisted Calderbank-Shor-Steane (CSS) codes. The protocol is provably secure from an eavesdropper, Eve, based on a result by Shor and Preskill [2].

The QKE protocol has a potential advantage over QKD, in that the original pair of classical codes considered need not have the dual-containing property. The cost is that the parties involved have to pre-share a secret key. The classical codes correspond to entanglement-assisted quantum error-correcting code (EAQECC). The EAQECC construction is described by the formalism given by Brun, Devetak and Hsieh [3].

In the CSS construction of Luo and Devetak's QKE protocol, a pair of classical linear codes with good error-correcting performance is needed. LDPC codes are classical linear codes that have sparse parity-check matrices, and many families of LDPC codes have been studied and claimed to give good performance (see, e.g., [4–10]). There were several recent studies on the performance of LDPC codes used for QKD [11, 12]. In this paper, LDPC codes constructed from finite geometry (FG) are considered [4, 10], and methods to incorporate them into the QKE protocol are proposed and explained. For simplicity, the quantum channel is modeled by the depolarizing channel. Given a tolerable bit error threshold ϵ for the generated keys, the goal is to search for codes that maximize the net key rate for given channel error parameters.

The paper is organized as follows. In section II, we first introduce the QKE protocol of Luo and Devetak. We then propose modifications to the post-processing steps to improve performance. In section III, we discuss families of LDPC codes generated by finite geometry. In section IV, we discuss simulation results using the improved QKE protocol from section II and the codes from section III, and we analyze their performance. In section V, we give conclusions and suggest possible work in the near future.

The one-dimensional vectors appearing in this paper should always be considered as column vectors. The vectors are denoted with <u>*underline*</u>, and the matrices are denoted with ***boldface***. The operations $+$ and $\oplus$ are defined respectively as component-wise addition and addition modulo 2.

II. QUANTUM KEY EXPANSION

The QKE protocol discussed in this paper is derived from the BB84 quantum key distribution protocol, using CSS codes for error correction and privacy amplification. The CSS code used for a BB84 QKD protocol is derived from a pair of "dual-containing" classical linear codes. Most pairs of classical codes do not satisfy this requirement, but such pairs can be found. The dual-containing property requires that $\boldsymbol{H}_1\boldsymbol{H}_2^T = \boldsymbol{0}$ to be satisfied, where $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ are the parity-check matrices of the two codes. The QKE protocol, however,

* kungchuh@usc.edu
† tbrun@usc.edu

does not require the pair of classical codes have the dual-containing restriction. The idea is to interpret the code as an entanglement-assisted code rather than a standard quantum code, and the cost is that the two parties involved must have a pre-shared secret key that is expanded by the protocol.

In subsection A, the structure of entanglement-assisted code will be introduced, as well as the notation that will be used throughout the paper. Subsection B reviews the steps of the QKE protocol proposed by Luo and Devetak [1]. In subsections C and D, we analyze the post-processing steps of the QKE protocol, and propose improvements. In subsection E, we summarize the improvements of subsection D and give a QKE protocol with enhanced performance compared to the original QKE protocol.

A. Code construction

This subsection summarizes the entanglement-assisted CSS code construction and the matrix structures involved. The notation mentioned here will be used throughout the later sections.

For $i = 1, 2$, let C_i be a classical $[n, k_i, d]$ code with parity-check matrix $\boldsymbol{H}_i$ of size $(n - k_i) \times n$. Based on the given pair of classical codes, an $[[n, k_1 + k_2 - n + c, d; c]]$ entanglement-assisted quantum CSS code can be constructed, where $c = \text{rank}(\boldsymbol{H}_1\boldsymbol{H}_2^T)$ is the number of ebits (or entangled pairs of qubits) needed. This code can protect $m = k_1 + k_2 - n + c$ qubits from error. After this process, we end up with two dual-containing classical codes C_1' and C_2' with "augmented" parity check matrices $\boldsymbol{H}_1'$ and $\boldsymbol{H}_2'$. The derivation of $\boldsymbol{H}_i'$ from $\boldsymbol{H}_i$ is as follows:

For a given pair of $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, there always exist nonsingular matrices $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ such that

$$\boldsymbol{T}_1\boldsymbol{H}_1\boldsymbol{H}_2^T\boldsymbol{T}_2^T = \begin{pmatrix} \boldsymbol{0}_{(n-k_1-c)\times(n-k_2-c)} & \boldsymbol{0}_{(n-k_1-c)\times c} \\ \boldsymbol{0}_{c\times(n-k_2-c)} & \boldsymbol{I}_c \end{pmatrix}. \tag{1}$$

$\boldsymbol{H}_i'$ can thus be constructed as follows to assure that the new codes satisfy the dual-containing property, $\boldsymbol{H}_1'\boldsymbol{H}_2'^T = \boldsymbol{0}$.

$$\boldsymbol{H}_i' = (\boldsymbol{T}_i\boldsymbol{H}_i \ \boldsymbol{J}_i), \text{ where } \boldsymbol{J}_i = \begin{pmatrix} \boldsymbol{0}_{(n-k_i-c)\times c} \\ \boldsymbol{I}_c \end{pmatrix}. \tag{2}$$

Suppose $\boldsymbol{H}_1'$ and $\boldsymbol{H}_2'$ are constructed. There exist binary matrices $\boldsymbol{E}_1$, $\boldsymbol{F}_1$, $\boldsymbol{E}_2$, and $\boldsymbol{F}_2$ such that the following four requirements are satisfied:

1. The rows of $\boldsymbol{H}_1'$ and $\boldsymbol{E}_1$ form a basis for C_2'.
2. The rows of $\boldsymbol{H}_2'$ and $\boldsymbol{E}_2$ form a basis for C_1'.
3. $\boldsymbol{N}_1 = \begin{pmatrix} \boldsymbol{H}_1' \\ \boldsymbol{E}_1 \\ \boldsymbol{F}_1 \end{pmatrix}$ and $\boldsymbol{N}_2 = \begin{pmatrix} \boldsymbol{F}_2 \\ \boldsymbol{E}_2 \\ \boldsymbol{H}_2' \end{pmatrix}$ are full rank matrices.
4. $\boldsymbol{N}_1\boldsymbol{N}_2^T = \boldsymbol{I}$.

The new parity-check matrices $\boldsymbol{H}_i'$ have more columns than the original $\boldsymbol{H}_i$. These columns correspond to additional qubits on the receiver's side. Before decoding, the sender (Alice) and the receiver (Bob) share c entangled pairs. Since Bob's half of these pairs do not pass through the channel, they are noise-free.

The syndrome of an error is defined as the error vector multiplied by the parity-check matrix of the code. For the code C_1' in our case, the syndrome corresponding to the error vector $\underline{e}$ is $\underline{s} = \boldsymbol{H}_1'\underline{e}$. The set of codewords of the code is the set of all vectors with zero syndromes.

The decoder for the LDPC codes considered in this paper is an SPA decoder [13] that identifies a probable error corresponding to each syndrome. Based on the decoder, the error set correctable by the code can be defined. For the code C_1' with parity-check matrix $\boldsymbol{H}_1'$, one may define such a set as $\mathcal{E}_1' = \{\boldsymbol{F}_2^T\underline{s} + \boldsymbol{E}_2^T\underline{\beta}(\underline{s}) + \boldsymbol{H}_2'^T\underline{\beta}'(\underline{s}) : \underline{s} \in \mathbb{Z}_2^{n-k_1}\}$, where $\underline{\beta}(\underline{.}) : \mathbb{Z}_2^{n-k_1} \to \mathbb{Z}_2^m$ and $\underline{\beta}'(\underline{.}) : \mathbb{Z}_2^{n-k_1} \to \mathbb{Z}_2^{n-k_2}$ are mappings fixed by the decoder. For every syndrome $\underline{s} \in \mathbb{Z}_2^{n-k_1}$, the decoder gives $\boldsymbol{F}_2^T\underline{s} + \boldsymbol{E}_2^T\underline{\beta}(\underline{s}) + \boldsymbol{H}_2'^T\underline{\beta}'(\underline{s})$ as the probable error. The receiver then corrects this error on the received codeword to retrieve the original message.

B. Luo and Devetak's quantum key expansion protocol

Let Alice and Bob be the sender and receiver utilizing the QKE protocol proposed in [1]. The steps of the protocol are:

1) Alice generates a binary string $\underline{a}$ consisted of $(2 + 3\delta)n$ random bits.

2) Alice generates another binary string $\underline{\alpha}$ consisted of $(2 + 3\delta)n$ random bits, and she prepares each bit in $\underline{a}$ in the Z or X basis according to the corresponding bit in $\underline{\alpha}$. For example, Alice may prepare the bit in $\underline{a}$ in the Z basis if the corresponding bit in $\underline{\alpha}$ is 0, and in the X basis otherwise.

3) Alice sends the prepared qubits to Bob.

4) Bob receives the qubits, and he generates a binary string $\underline{\gamma}$ consisting of $(2 + 3\delta)n$ random bits. Bob then uses $\underline{\gamma}$ to determine in which bases to measure the received qubits. To be consistent with the example in 2), Bob measures the received qubit in the Z basis if the corresponding bit in $\underline{\gamma}$ is 0 and measures in the X basis otherwise. Let the resulting bit string be $\underline{b}$.

5) Alice announces $\underline{\alpha}$, and Bob discards the bits in $\underline{b}$ where the corresponding bits in $\underline{\gamma}$ and $\underline{\alpha}$ don't match, that is, the bit locations where they prepare and measure in different bases. Bob announces which bits he discards. With high probability, there are at least $(1+\delta)n$ bits left; if not, they abort and restart the protocol.

6) Alice randomly chooses n bits and announces the bit locations for Bob to extract the corresponding bits. Let Alice's resulting string be $\underline{\hat{a}}$, and Bob's be $\underline{\hat{b}}$. There are at least $n\delta$ pairs of bits left, and those pairs are used for

channel estimation. Alice and Bob announce those bits to each other and count the fraction that do not match. If there are too many errors, they abort and restart the protocol.

7) Alice attaches the length-c pre-shared bit string $\underline{\kappa}$ to $\underline{\hat{a}}$. She first computes $\underline{s}_A = \boldsymbol{H}_1' \begin{pmatrix} \underline{\hat{a}} \\ \underline{\kappa} \end{pmatrix}$ and announces it to Bob. She then computes her part of the generated key, $\underline{k_A} = \boldsymbol{E_1} \begin{pmatrix} \underline{\hat{a}} \\ \underline{\kappa} \end{pmatrix}$.

8) Bob computes $\underline{s}_B = \boldsymbol{H}_1' \begin{pmatrix} \underline{\hat{b}} \\ \underline{\kappa} \end{pmatrix}$, and his part of the generated key is $\underline{k}_B = \boldsymbol{E}_1 \begin{pmatrix} \underline{\hat{b}} \\ \underline{\kappa} \end{pmatrix} \oplus \underline{\beta}(\underline{s}_A \oplus \underline{s}_B)$.

C. Analysis of QKE post-processing

Consider the procedure of Luo and Devetak's QKE protocol formalized in the previous subsection. The error correction is performed at the last step 8) where Bob computes $\underline{\beta}(\underline{s}_A \oplus \underline{s}_B)$. In this case, $\underline{s}_A \oplus \underline{s}_B$ is the syndrome that initializes the decoding. To understand how the function $\underline{\beta}(\underline{\cdot})$ is computed, we need to examine its definition and the matrix structure of the code.

Suppose we start with two LDPC codes with parity-check matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ of sizes $(n-k_1)\times n$ and $(n-k_2)\times n$, and $c = \text{rank}(\boldsymbol{H}_1\boldsymbol{H}_2^T)$. The formalism in subsection A gives two $(n+c)\times(n+c)$ full rank matrices $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$, each formed by 3 block-matrices $\boldsymbol{H}_i'$, $\boldsymbol{E}_i$, and $\boldsymbol{F}_i$ of sizes $(n-k_i)\times(n+c)$, $(k_1+k_2-n+c)\times(n+c)$, and $(n-k_{(1+i \bmod 2)})\times(n+c)$ respectively. $\boldsymbol{H}_1'$ and $\boldsymbol{H}_2'$ are defined as the parity check matrices of the newly formed entanglement-assisted CSS code. Note that the two new parity-check matrices need not be low-density and thus the performance will be poor if one uses them to run the SPA decoder. However, as seen in subsection A, since the matrix operations transforming $\boldsymbol{H}_i$ to $\boldsymbol{H}_i'$ are reversible, the error syndrome with respect to the original parity-check matrix $\boldsymbol{H}_i$ can be retrieved by doing inverse matrix operations on the corresponding syndrome with respect to $\boldsymbol{H}_i'$. That is, given a syndrome corresponding to $\boldsymbol{H}_i'$, we can find the corresponding syndrome for $\boldsymbol{H}_i$. As a result, the errors can be decoded by the SPA decoder with LDPC matrix $\boldsymbol{H}_i$. The details follow.

The function $\underline{\beta}(\underline{\cdot})$, which includes the process of error correction, comes into the picture when the error set $\mathcal{E}_1$ correctable by the code $\boldsymbol{H}_1'$ is defined. Recall from subsection A, $\mathcal{E}_1 = \{\boldsymbol{F}_2^T\underline{s} + \boldsymbol{E}_2^T\underline{\beta}(\underline{s}) + \boldsymbol{H}_2'^T\underline{\beta}'(\underline{s}) : \underline{s} \in \mathbb{Z}_2^{n-k_1}\}$. Since the matrix $\boldsymbol{N}_2$ formed by $\boldsymbol{H}_2'$, $\boldsymbol{E}_2$, and $\boldsymbol{F}_2$ is a full rank matrix in $\mathbb{Z}_2$, the error string corresponding to a particular syndrome $\underline{s}$ can be retrieved by the following steps:

i) Compute $\underline{s}' = \boldsymbol{T}_1^{-1}\underline{s}$.

ii) Run the SPA decoder using the original LDPC matrix $\boldsymbol{H}_1$ with the syndrome $\underline{s}'$. The decoded string is the estimated error, and we denote it by $\underline{\hat{e}}$.

iii) Attach c 0's to $\underline{\hat{e}}$ and compute $\underline{\beta}(\underline{s}) = \boldsymbol{E}_1 \begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix}$.

In the above steps i) and ii), the error message can be decoded using $\boldsymbol{H}_1$ instead of $\boldsymbol{H}_1'$ since the last c bits of the message are pre-shared by Alice and Bob, and thus the error message from those bits should always be a string of 0's. The syndrome is then totally determined by the first n bits of the error message. This allows us to use the original low-density parity-check matrices for decoding and thus the error-correcting performance is maintained.

The last step may not be trivial, and we explain it in the following. Using our notation, if $\begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix}$ is correctable by $\boldsymbol{H}_1'$ with syndrome $\underline{s}$, it is in the set $\mathcal{E}_1$ and can be written in the form

$$\begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix} = \boldsymbol{N}_2^T \begin{pmatrix} \underline{s} \\ \underline{\beta}(\underline{s}) \\ \underline{\beta}'(\underline{s}) \end{pmatrix}. \tag{3}$$

Since $\boldsymbol{N}_1\boldsymbol{N}_2^T = \boldsymbol{I}$, it is obvious that $\boldsymbol{N}_2^T = \boldsymbol{N}_1^{-1}$. $\boldsymbol{N}_1$ can then be multiplied to both sides of the above equation. As a result,

$$\begin{pmatrix} \underline{s} \\ \underline{\beta}(\underline{s}) \\ \underline{\beta}'(\underline{s}) \end{pmatrix} = \boldsymbol{N}_1 \begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}_1' \\ \boldsymbol{E}_1 \\ \boldsymbol{F}_1 \end{pmatrix} \begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix}. \tag{4}$$

It should now be clear that step iii) is valid.

D. Improving QKE post-processing

A very important observation based on our simulations is that in the cases where the channel error rates are not small, the bit error rates of the resulting keys are significant whenever the estimated errors $\begin{pmatrix} \underline{\hat{e}} \\ \underline{0}_{c\times 1} \end{pmatrix}$ are erroneous. Specifically, the bit error rates of the keys are about half the block error rates for sufficiently large channel error probabilities. Since $\underline{\beta}(\underline{\cdot})$ is equivalent to multiplying by a matrix, $\boldsymbol{E}_1$, this observation implies that $\boldsymbol{E}_1$ is generally not sparse. Given a block error, it is likely that each row of $\boldsymbol{E}_1$ and the block error have overlapping non-zero elements, which on average contributes to a significant number of errors in the key. In other words, when a block error occurs the resulting key is almost totally randomized.

From the observation above, we can apply two useful improvements to the protocol.

Improvement 1 is to check the syndrome following the decoder's output. This allows the detection of not-yet-converged messages from the SPA decoder. These

messages must have block errors. Aborting the protocol after detecting those erroneous messages greatly improves the error performance of the generated key, at the cost of modestly reducing the key rate, since the information sent through the channel in the prior stages is wasted.

Improvement 2 is to check the generated keys directly. Let the block error rate and bit error rate of the generated keys be denoted by R_{blk} and R_{bit}. Since block errors of the keys result in a large fraction of the bits being erroneous in each block, checking several randomly chosen bits allows a large probability of detecting those block errors. Let us assume the relationship $R_{bit} = qR_{blk}$, such that, on average, a block error yields a bit error rate of q. Suppose each time the protocol is processed, a number of bits μ are chosen randomly from the key, and are used for a check between the sender and the receiver. The bit error rate of the generated key, $\hat{R}_{bit}$, can then be calculated as

$$\hat{R}_{bit} = R_{bit}\frac{(1-q)^{\mu}}{1-R_{blk}+(1-q)^{\mu}R_{blk}} \equiv R_{bit}f. \quad (5)$$

The bit error rate is scaled by the factor f. For fixed R_{blk}, f decreases dramatically as μ increases. This means that not many bits need be checked to greatly improve the error performance of the key. To determine μ, we find the smallest μ satisfying $\hat{R}_{bit} < \epsilon$, where ϵ is the desired threshold for the bit error rate of the final key. That is,

$$\mu = \begin{cases} \lceil log_{(1-q)}(\frac{\epsilon(1-R_{blk})}{(q-\epsilon)R_{blk}})\rceil & \text{if } q > \epsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Since those randomly chosen μ bits from the key are revealed, the tradeoff in using this method would be to reduce key rate by an amount $\frac{\mu}{n}$.

A problem arises here, in that the pre-shared key bits are consumed even if the protocol fails, which could even result in the net key rate being negative. However, there is a way to get around this problem.

In the original QKE protocol, Alice announces to Bob the message $\underline{s}_A = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{a}}\\ \underline{\kappa}\end{pmatrix}$, and Bob corrects the errors using the syndrome $\underline{s} = \underline{s}_A \oplus \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{b}}\\ \underline{\kappa}\end{pmatrix} = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{a}}\oplus\hat{\underline{b}}\\ \underline{0}\end{pmatrix}$. This syndrome can also be computed by Bob if Alice sends the message $\hat{\underline{s}}_A = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{a}}\\ \underline{0}\end{pmatrix}$ instead. In this case, Bob just computes $\underline{s} = \hat{\underline{s}}_A \oplus \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{b}}\\ \underline{0}\end{pmatrix} = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{a}}\oplus\hat{\underline{b}}\\ \underline{0}\end{pmatrix}$.

Thus, instead of comparing the keys $\underline{k}_A = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{a}}\\ \underline{\kappa}\end{pmatrix}$ and $\underline{k}_B = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{b}}\\ \underline{\kappa}\end{pmatrix} \oplus \underline{\beta}(\underline{s}_A \oplus \underline{s}_B)$ and consuming the pre-shared key $\underline{\kappa}$, it is sufficient for the two parties to compare $\hat{\underline{k}}_A = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{a}}\\ \underline{0}\end{pmatrix}$ and $\hat{\underline{k}}_B = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{b}}\\ \underline{0}\end{pmatrix} \oplus \underline{\beta}(\hat{\underline{s}}_A \oplus \hat{\underline{s}}_B)$. In this way, we can postpone the consumption of the pre-shared keys until after the check is performed. Note that, Alice and Bob must discard the bits from the final key corresponding to the ones they compare, since information about those bits is publicly revealed.

E. Summary of the improved QKE protocol

In this subsection, we will combine the two improvements from the previous subsection and assess the improved performance of the QKE protocol. We consider the case where **Improvement 1** is performed first, and then **Improvement 2** is performed if the check in **Improvement 1** is successful.

Let p_1 be the failure rate of the check in **Improvement 1**. Conditioned on passing the check in **Improvement 1**, let p_2 be the rate of bit errors in the generated keys followed by the remaining block errors. Also, let R_{blk} be the block error rate of the LDPC code and ϵ be the error threshold that is desired for QKE. The values, R_{blk}, p_1 and p_2, can be determined by simulation. After **Improvement 2** is performed, the bit error rate of the generated key, $\hat{R}_{bit}$, can then be calculated:

$$\hat{R}_{bit} = p_2\frac{(1-p_2)^{\mu}(R_{blk}-p_1)}{1-R_{blk}+(1-p_2)^{\mu}(R_{blk}-p_1)}. \quad (7)$$

To determine μ, we find the smallest μ satisfying $\hat{R}_{bit} < \epsilon$. That is,

$$\mu = \begin{cases} \lceil log_{(1-p_2)}(\frac{\epsilon(1-R_{blk})}{(p_2-\epsilon)(R_{blk}-p_1)})\rceil & \text{if } p_2 > \epsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

.

We now outline the improved QKE protocol. Referring to the original QKE protocol in subsection B, the procedure up to step 6) will be the same. The steps beyond 7) are modified as follows:

7) Alice computes $\hat{\underline{s}}_A = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{a}}\\ \underline{0}\end{pmatrix}$ and announces it to Bob.

8) Bob first computes $\hat{\underline{s}}_B = \boldsymbol{H}_1'\begin{pmatrix}\hat{\underline{b}}\\ \underline{0}\end{pmatrix}$, and then he runs the SPA decoder using the original LDPC matrix $\boldsymbol{H}_1$ with the syndrome $\underline{s}' = \boldsymbol{T}_1^{-1}(\hat{\underline{s}}_A \oplus \hat{\underline{s}}_B)$. Let the decoded error string be $\hat{\underline{e}}$.

9) Bob checks if $\boldsymbol{H}_1\hat{\underline{e}} \oplus \underline{s}'$ is the all-zero string. If not, the protocol is aborted and they start over. This is a result of **Improvement 1**.

10) Alice randomly chooses μ bits from $\hat{\underline{k}}_A = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{a}}\\ \underline{0}\end{pmatrix}$ and announces them to Bob. Bob checks if the corresponding bits from $\hat{\underline{k}}_B = \boldsymbol{E}_1\begin{pmatrix}\hat{\underline{b}}\oplus\hat{\underline{e}}\\ \underline{0}\end{pmatrix}$ match the ones sent by Alice. If the strings do not completely match, the protocol is aborted and they start over. This is a result of **Improvement 2**.

11) Alice computes her part of the generated key as $\underline{k}_A = \hat{\underline{k}}_A \oplus \boldsymbol{E}_1 \begin{pmatrix} 0 \\ \underline{\kappa} \end{pmatrix}$, excluding the μ bits corresponding to the ones they have compared in the previous step. Bob also computes his part of the generated key as $\underline{k}_B = \hat{\underline{k}}_B \oplus \boldsymbol{E}_1 \begin{pmatrix} 0 \\ \underline{\kappa} \end{pmatrix}$, excluding the μ bits similarly.

The pre-shared key is only used in the last step. Therefore, the pre-shared key will not be consumed if the protocol is aborted in steps 10) or 11). The net key rate of this improved QKE protocol is

$$R_{net} = (1 - R_{blk} + (1-p_2)^{\mu} (R_{blk} - p_1)) \frac{m - c - \mu}{n}. \tag{9}$$

We will see how well this does in simulations below.

III. FINITE GEOMETRY LDPC CODES

Finite geometry (FG) LDPC codes were formalized by Kou, Lin and Fossorier [4]. There are four families of FG LDPC codes: type-1 Euclidean geometry (EG1) LDPC codes, type-2 Euclidean geometry (EG2) LDPC codes, type-1 projective geometry (PG1) LDPC codes, and type-2 projective geometry (PG2) LDPC codes. These classical FG LDPC codes were used by Hsieh, Yen and Hsu to construct EAQECCs with good performance that use relatively little entanglement [10]. In this section, we briefly restate the results from [4] and [10] and introduce the construction of FG LDPC codes.

A. Euclidean geometry (EG) LDPC codes

Let $\mathrm{EG}(p, 2^s)$ be an p-dimensional Euclidean geometry over the Galois field $\mathrm{GF}(2^s)$, where $p, s \in \mathbb{N}$. This geometry consists of 2^{ps} points, where each is an p-tuple over $\mathrm{GF}(2^s)$. The all-zero p-tuple is defined as the origin. Those points form an p-dimensional vector space over $\mathrm{GF}(2^s)$. A line in $\mathrm{EG}(p, 2^s)$ is a coset of a one-dimensional subspace of $\mathrm{EG}(p, 2^s)$, and each line consists of 2^s points. There are $2^{(p-1)s}(2^{ps}-1)/(2^s-1)$ lines. Each line has $2^{(p-1)s}-1$ lines parallel to it. Each point is intersected by $(2^{ps}-1)/(2^s-1)$ lines.

Let $\mathrm{GF}(2^{ps})$ be the extension field of $\mathrm{GF}(2^s)$. Each element in $\mathrm{GF}(2^{ps})$ can be represented as an p-tuple over $\mathrm{GF}(2^s)$, and hence a point in $\mathrm{EG}(p, 2^s)$. Therefore, $\mathrm{GF}(2^{ps})$ may be regarded as the Euclidean geometry $\mathrm{EG}(p, 2^s)$. Let α be a primitive element of $\mathrm{GF}(2^{ps})$. Then $0, \alpha^0, \alpha^1, \alpha^1, ..., \alpha^{2^{ps}-2}$ represent the 2^{ps} points of $\mathrm{EG}(p, 2^s)$.

Let $\boldsymbol{H}_{EG1}(p, s)$ be a matrix over $\mathrm{GF}(2)$. The rows of $\boldsymbol{H}_{EG1}(p, s)$ are the incidence vectors of all the lines in $\mathrm{EG}(p, 2^s)$ not passing through the origin. The columns of $\boldsymbol{H}_{EG1}(p, s)$ are the $2^{ps}-1$ non-origin points of $\mathrm{EG}(p, 2^s)$, and the ith column corresponds to the point α^{i-1}. Then $\boldsymbol{H}_{EG1}(p, s)$ consists of $n = 2^{ps}-1$ columns and $J = (2^{(p-1)s}-1)(2^{ps}-1)/(2^s-1)$ rows, and it has the following structure:

1. Each row has weight $\rho_r = 2^s$.
2. Each column has weight $\rho_c = (2^{ps}-1)/(2^s-1)-1$.
3. Any two columns have at most one "1-component" in common.
4. Any two rows have at most one "1-component" in common.

The density of $\boldsymbol{H}_{EG1}(p, s)$ is $2^s/(2^{ps}-1)$, which is small for p or s large. Then $\boldsymbol{H}_{EG1}(p, s)$ is a low-density matrix.

The LDPC code with parity-check matrix $\boldsymbol{H}_{EG1}(p, s)$ is called a type-1 Euclidean geometry LDPC code, and we denote it by $EG1(p, s)$.

Let $\boldsymbol{H}_{EG2}(p, s) = \boldsymbol{H}_{EG1}(p, s)^T$. Then $\boldsymbol{H}_{EG2}(p, s)$ is a matrix with $2^{ps}-1$ rows and $(2^{(p-1)s}-1)(2^{ps}-1)/(2^s-1)$ columns. The rows of $\boldsymbol{H}_{EG2}(p, s)$ are the non-origin points of $\mathrm{EG}(p, 2^s)$, and the columns are the lines in $\mathrm{EG}(p, 2^s)$ not passing through the origin, and it has the following structure:

1. Each row has weight $\rho_r = (2^{ps}-1)/(2^s-1)-1$.
2. Each column has weight $\rho_c = 2^s$.
3. Any two columns have at most one "1-component" in common.
4. Any two rows have at most one "1-component" in common.

The LDPC code with parity-check matrix $\boldsymbol{H}_{EG2}(p, s)$ is called a type-2 Euclidean geometry LDPC code, and we denote it by $EG2(p, s)$.

B. Projective geometry (PG) LDPC codes

Let $\mathrm{GF}(2^{(p+1)s})$ be the extension field of $\mathrm{GF}(2^s)$. Let α be a primitive element of $\mathrm{GF}(2^{(p+1)s})$. Let $n = (2^{(p+1)s}-1)/(2^s-1)$ and $\eta = \alpha^n$. Then η has order 2^s-1, and the 2^s elements $0, \eta^0, \eta^1, \eta^2, ..., \eta^{2^s-2}$ form all the elements of $\mathrm{GF}(2^s)$. Consider the set $\{\alpha^0, \alpha^1, \alpha^2, ..., \alpha^{n-1}\}$, and partition the non-zero elements of $\mathrm{GF}(2^{(m+1)s})$ into n disjoint subsets $\{\alpha^i, \eta\alpha^i, \eta^2\alpha^i, ..., \eta^{2^s-2}\alpha^i\}$, for $i \in \{0, 1, ..., n-1\}$. Each such set is represented by its first element (α^i), for $i \in \{0, 1, ..., n-1\}$.

If each element in $\mathrm{GF}(2^{(p+1)s})$ is represented as a $(p+1)$-tuple over $\mathrm{GF}(2^s)$, then (α^i) consists of 2^s-1 $(p+1)$-tuples over $\mathrm{GF}(2^s)$. The $(p+1)$-tuple over $\mathrm{GF}(2^s)$ that represents (α^i) can be regarded as a point in a finite geometry over $\mathrm{GF}(2^s)$. Then the points $(\alpha^0), (\alpha^1), (\alpha^2), ..., (\alpha^{n-1})$ form a p-dimensional projective geometry over $\mathrm{GF}(2^s)$, denoted $\mathrm{PG}(p, 2^s)$. (Note that a projective geometry does not have an origin.)

Let $\boldsymbol{H}_{PG1}(p, s)$ be a matrix over $\mathrm{GF}(2)$. The rows of $\boldsymbol{H}_{PG1}(p, s)$ are the incidence vectors of all the lines in $\mathrm{PG}(p, 2^s)$. The columns of $\boldsymbol{H}_{PG1}(p, s)$ are the n points of $\mathrm{PG}(p, 2^s)$, and the ith column corresponds to the point (α^{i-1}). Then $\boldsymbol{H}_{PG1}(p, s)$ consists of $n = (2^{(p+1)s}-1)/(2^s-1)$ columns and $J = (2^{ps} + ... + 2^s + 1)(2^{(p-1)s} + ... + 2^s + 1)/(2^s+1)$ rows, and it has the following structure:

1. Each row has weight $\rho_r = 2^s + 1$.

2. Each column has weight $\rho_c = (2^{ps} - 1)/(2^s - 1)$.

3. Any two columns have at most one "1-component" in common.

4. Any two rows have at most one "1-component" in common.

The density of $\boldsymbol{H}_{PG1}(p, s)$ is $(2^{2s} - 1)/(2^{(p+1)s} - 1)$, which is small for p or s large. Then $\boldsymbol{H}_{PG1}(p, s)$ is a low-density matrix.

The LDPC code with parity-check matrix $\boldsymbol{H}_{PG1}(p, s)$ is called a type-1 projective geometry LDPC code, and we denote it by $PG1(p, s)$.

Let $\boldsymbol{H}_{PG2}(p, s) = \boldsymbol{H}_{PG1}(p, s)^T$. Then $\boldsymbol{H}_{PG2}(p, s)$ is a matrix with $(2^{(p+1)s} - 1)/(2^s - 1)$ rows and $(2^{ps} + ... + 2^s + 1)(2^{(p-1)s} + ... + 2^s + 1)/(2^s + 1)$ columns. The rows of $\boldsymbol{H}_{PG2}(p, s)$ are the points of $\mathrm{PG}(p, 2^s)$, and the columns are the lines in $\mathrm{PG}(p, 2^s)$, and it has the following structure:

1. Each row has weight $\rho_r = (2^{ps} - 1)/(2^s - 1)$.

2. Each column has weight $\rho_c = 2^s + 1$.

3. Any two columns have at most one "1-component" in common.

4. Any two rows have at most one "1-component" in common.

The LDPC code with parity-check matrix $\boldsymbol{H}_{PG2}(p, s)$ is called a type-2 projective geometry LDPC code, and we denote it by $PG2(p, s)$.

C. Extension of finite geometry LDPC codes by column and row splitting

A finite geometry LDPC code with n columns and J rows can be extended by splitting each column of its parity-check matrix H into multiple columns. If the splitting is done properly, very good extended finite geometry LDPC codes can be obtained.

Let $\underline{g}_1, \underline{g}_2, ..., \underline{g}_n$ be the columns of $\boldsymbol{H}$. Let c_{sp} be the column splitting factor, $c_{sp} \in \{1, 2, ..., \rho_c\}$. Then the column splitting can be done by splitting each $\underline{g}_i$ into c_{sp} columns $\underline{g}_{i,1}, \underline{g}_{i,2}, ..., \underline{g}_{i,c_{sp}}$, and distribute the ones of the original column among the new columns accordingly. So that the columns $\underline{g}_{i,1}, \underline{g}_{i,2}, ..., \underline{g}_{i,\rho_c - c_{sp}\lfloor \frac{\rho_c}{c_{sp}} \rfloor}$ have weights $\frac{\rho_c}{c_{sp}} + 1$, and the other columns have weights $\frac{\rho_c}{c_{sp}}$.

After column splitting, we can proceed with row splitting, that is, determine a row splitting factor $r_{sp} \in \{1, 2, ..., \rho_r\}$ and follow similarly the process of column splitting.

We denote $EG1(p, s, c_{sp}, r_{sp})$ as the LDPC code constructed by an $EG1(p, s)$ LDPC code with column and row splitting factors c_{sp} and r_{sp}. The codes $EG2(p, s, c_{sp}, r_{sp})$, $PG1(p, s, c_{sp}, r_{sp})$, $PG2(p, s, c_{sp}, r_{sp})$ are defined similarly.

IV. SIMULATION RESULTS

In this section, we provide simulation results of our QKE protocol with FG codes. We use the same LDPC code for both C_1 and C_2 in constructing the entanglement-assisted CSS code for our QKE protocol. The channel for quantum communication is assumed to be a depolarizing channel, and the channel error probability P_e in the simulation corresponds to that of the equivalent classical binary-symmetric channel (BSC). We use Monte Carlo simulation with a sample size of $200,000$ for each P_e. We allow the SPA decoder to iterate a maximum of 100 times.

The first group of FG codes we demonstrate is the family of 2-dimensional type-1 PG LDPC codes without splitting. These $PG1(2, s, 1, 1)$ codes require only 1 bit of entanglement per codeword as was proven by Hsieh et al. [10] Therefore, it is possible to implement QKE with only 1 bit of pre-shared secret key. The other families of FG codes we consider are the $EG1(2, 5, c_{sp}, r_{sp})$ and $PG1(2, 5, c_{sp}, r_{sp})$ codes.

A. $PG1(2, s, 1, 1)$ codes

For the $PG1(2, s, 1, 1)$ codes that require only 1 bit of pre-shared key, we consider the equivalent BSC bit error probabilities ranging from 0% to 8% in steps of 0.5%. Let $[[n, m; c]]$ be the parameters of the entanglement-assisted code, and R_{net} be the original net key rate of QKE using that code; that is, $R_{net} = \frac{m-c}{n}$. This means that the QKE protocol expands a key of length c to a key of length m. Table I demonstrates all possible $PG1(2, s, 1, 1)$ codes that have block length $n \leq 10000$.

TABLE I. List of $PG1(2, s, 1, 1)$ codes that have block length $n \leq 10000$.

s	$[[n, m; c]]$	R_{net}
2	$[[21, 2; 1]]$	0.0476
3	$[[73, 18; 1]]$	0.2329
4	$[[273, 110; 1]]$	0.3993
5	$[[1057, 570; 1]]$	0.5383
6	$[[4161, 2702; 1]]$	0.6491

In Fig. 1, we show the QKE performance of the original protocol, in terms of bit error rate, of the codes from Table I. In Fig. 2, we set the generated keys' bit error threshold to $\epsilon = 10^{-6}$, and simulate QKE with the improved QKE protocol from section II. We present the performance, in terms of net key rate, using the codes from Table I.

The results from Fig. 2 show that it is possible to use just 1 bit of pre-shared key for QKE even when the channel is moderately noisy up to a bit error probability of 8%. In addition, the codes that are considered have reasonable block sizes, and therefore the QKE protocol can be efficiently implemented.

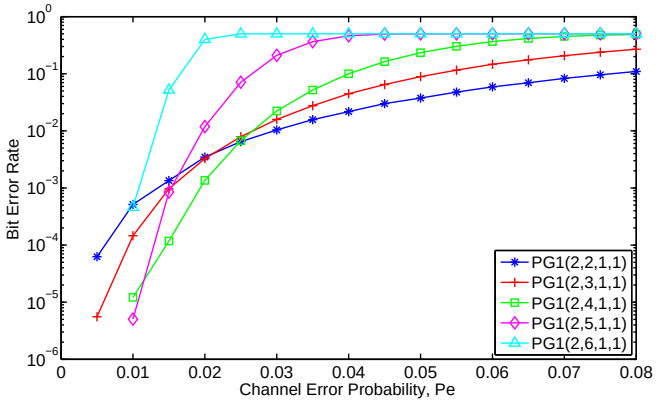

FIG. 1. (Color online) Bit error rate of the keys generated by the original QKE protocol with the $PG1(2, s, 1, 1)$ codes from Table I.

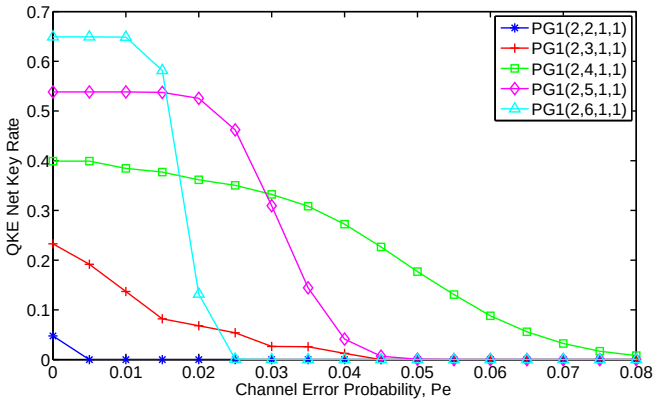

FIG. 2. (Color online) Net key rate of the improved QKE protocol with the $PG1(2, 5, c_{sp}, r_{sp})$ codes from Table I. The error threshold is set to $\epsilon = 10^{-6}$.

B. $EG1(2, 5, c_{sp}, r_{sp})$ and $PG1(2, 5, c_{sp}, r_{sp})$ codes

For the simulation of the $EG1(2, 5, c_{sp}, r_{sp})$ and $PG1(2, 5, c_{sp}, r_{sp})$ codes, we consider the equivalent BSC bit error probabilities ranging from 2% to 8% in steps of 0.5%. Since many codes perform well when P_e is small, we are mostly interested in codes that have good performance for higher P_e, such as might occur in realistic experiments. For a code to serve the purpose of performing key "expansion," one requires R_{net} to be positive. Table II lists all possible $EG1(2, 5, c_{sp}, r_{sp})$ codes with positive R_{net} that have block length $n \leq 11000$. In Fig. 3, we show the QKE performance of the original protocol, in terms of bit error rate, of some codes from Table II.

In Fig. 4, we set the generated keys' bit error threshold to $\epsilon = 10^{-6}$, and simulate QKE with the improved QKE protocol from section II. We present the performance, in terms of net key rate, using some codes from Table II.

Table III lists all possible $PG1(2, 5, c_{sp}, r_{sp})$ codes with positive R_{net} that have block length $n \leq 11000$. In Fig. 5, we present the QKE performance of the original protocol, in terms of bit error rate, of some codes from Table III.

In Fig. 6, we set the generated keys' bit error threshold to $\epsilon = 10^{-6}$ and simulate QKE with the improved QKE protocol proposed in section II. We present the performance, in terms of net key rate, using some codes from Table III.

Note that for channel error rates less than 2%, we may consider the code $PG1(2, 5, 9, 2)$, which has a net key rate of about 0.5556. Considering channel error rates much lower than 2%, we can use other codes in the family which have even larger net key rates.

TABLE II. List of $EG1(2, 5, c_{sp}, r_{sp})$ codes with positive net key rates that have block length $n \leq 11000$.

$[[n, m; c]]$	c_{sp}	r_{sp}	R_{net}
$[[1023, 571; 32]]$	1	1	0.5269
$[[2046, 452; 450]]$	2	1	0.0010
$[[3069, 2045; 1022]]$	3	1	0.3333
$[[4092, 3068; 1020]]$	4	1	0.5005
$[[4092, 2038; 2034]]$	4	2	0.0010
$[[5115, 4091; 1022]]$	5	1	0.6000
$[[5115, 3067; 2044]]$	5	2	0.2000
$[[6138, 5114; 1022]]$	6	1	0.6667
$[[6138, 4090; 2044]]$	6	2	0.3333
$[[7161, 6137; 1022]]$	7	1	0.7143
$[[7161, 5115; 2046]]$	7	2	0.4286
$[[7161, 4092; 3069]]$	7	3	0.1429
$[[8184, 7152; 1012]]$	8	1	0.7502
$[[8184, 6138; 2042]]$	8	2	0.5005
$[[8184, 5115; 3067]]$	8	3	0.2502
$[[8184, 4094; 4082]]$	8	4	0.0015
$[[9207, 8181; 1020]]$	9	1	0.7778
$[[9207, 7161; 2046]]$	9	2	0.5556
$[[9207, 6134; 3065]]$	9	3	0.3333
$[[9207, 5115; 4092]]$	9	4	0.1111
$[[10230, 9202; 1018]]$	10	1	0.8000
$[[10230, 8182; 2044]]$	10	2	0.6000
$[[10230, 7160; 3068]]$	10	3	0.4000
$[[10230, 6132; 4086]]$	10	4	0.2000

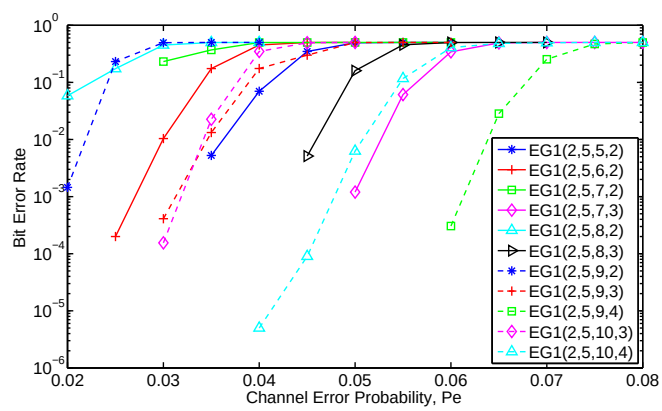

FIG. 3. (Color online) Bit error rate of the keys generated by the original QKE protocol with selected codes from $EG1(2, 5, c_{sp}, r_{sp})$.

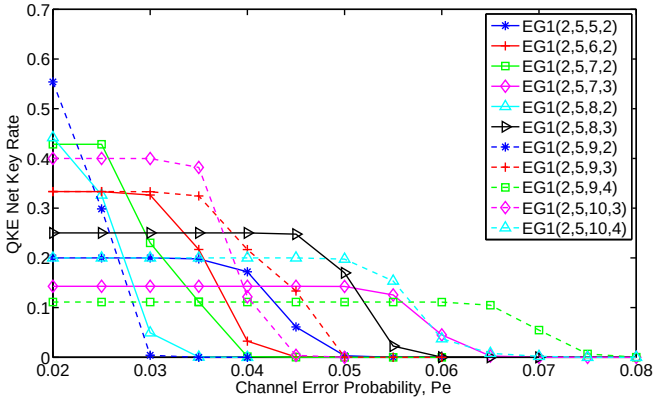

FIG. 4. (Color online) Net key rate of the improved QKE protocol with selected codes from $EG1(2,5,c_{sp},r_{sp})$ and error threshold $\epsilon = 10^{-6}$.

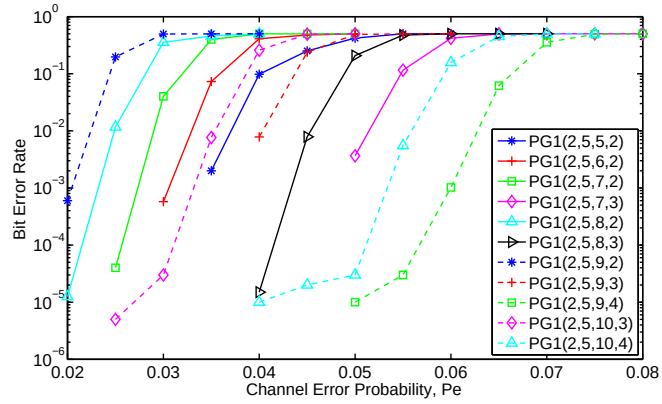

FIG. 5. (Color online) Bit error rate of the keys generated by the original QKE protocol with selected codes from $PG1(2,5,c_{sp},r_{sp})$.

TABLE III. List of $PG1(2,5,c_{sp},r_{sp})$ codes with positive net key rates that have block length $n \leq 11000$.

$[[n,m;c]]$	c_{sp}	r_{sp}	R_{net}
$[[1057,570;1]]$	1	1	0.5383
$[[2114,490;488]]$	2	1	0.0009
$[[3171,2112;1055]]$	3	1	0.3333
$[[4228,3172;1056]]$	4	1	0.5005
$[[4228,2114;2112]]$	4	2	0.0005
$[[5285,4227;1056]]$	5	1	0.6000
$[[5285,3171;2114]]$	5	2	0.2000
$[[6342,5284;1056]]$	6	1	0.6667
$[[6342,4228;2114]]$	6	2	0.3333
$[[7399,6341;1056]]$	7	1	0.7143
$[[7399,5285;2114]]$	7	2	0.4286
$[[7399,4227;3170]]$	7	3	0.1429
$[[8456,7399;1055]]$	8	1	0.7502
$[[8456,6342;2112]]$	8	2	0.5002
$[[8456,5286;3170]]$	8	3	0.2502
$[[8456,4229;4227]]$	8	4	0.0002
$[[9513,8455;1056]]$	9	1	0.7778
$[[9513,7399;2114]]$	9	2	0.5556
$[[9513,6342;3171]]$	9	3	0.3333
$[[9513,5284;4227]]$	9	4	0.1111
$[[10570,9511;1055]]$	10	1	0.8000
$[[10570,8456;2114]]$	10	2	0.6000
$[[10570,7399;3171]]$	10	3	0.4000
$[[10570,6342;4228]]$	10	4	0.2000

Finally, in Fig. 7, we set the generated keys' bit error threshold to $\epsilon = 10^{-6}$, and we present the QKE net rate using the codes from Tables I, II and III that perform the best in each channel error region within 2% to 8%. As can be seen, quite reasonable key rates can be achieved even for error probabilities above 7%.

It is worthwhile comparing our results to the recent work by Elkouss, Leverrier, Alléaume and Boutros [12]. In their work, a set of 9 irregular LDPC codes were found for QKD based on the BB84 protocol. With a bit error rate threshold of the generated keys on the same order as

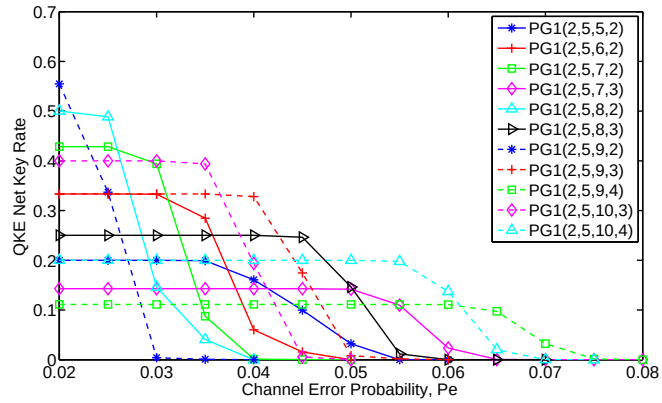

FIG. 6. (Color online) Net key rate of the improved QKE protocol with selected codes from $PG1(2,5,c_{sp},r_{sp})$ and error threshold $\epsilon = 10^{-6}$.

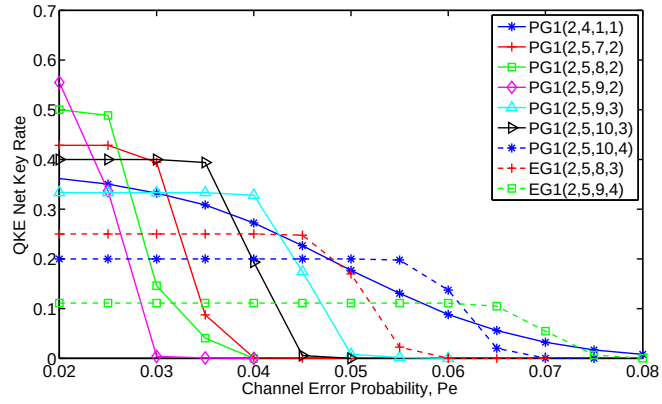

FIG. 7. (Color online) Net key rate of the improved QKE protocol with selected codes from both $EG1(2,5,c_{sp},r_{sp})$ and $PG1(2,5,c_{sp},r_{sp})$ that perform well in the various channel error regions.

ours (1.5×10^{-6} in their case), their net key rate performance exceeds ours by roughly $15 - 20\%$ over the same channel error regions. However, this is not too surprising, since they consider LDPC codes with very large block sizes (on the order of 10^6 bits), while ours have much more modest block sizes (on the order of 10^3). We believe the sizes of our codes are reasonable for practical use. Given much greater computing resources for post-processing, it should be easy to construct very large codes in our family of LDPC codes that would have better net key rates.

V. CONCLUSION

In this paper, we have proposed a protocol for QKE that is an improved version of the protocol proposed by Luo and Devetak. The modifications are done to filter out block errors, which allows us to greatly reduce the bit error rate of QKE with only a small reduction in the net key rate. In addition, we have studied a family of LDPC codes based on finite geometry that are capable of protecting the QKE protocol from errors even when the channel is moderately noisy. The figures in the previous section show clearly which codes one should choose to efficiently expand the keys.

In the near future we will investigate other families of codes for this QKE protocol. The LDPC codes generated by finite geometry are a rich family. Besides the family of FG codes constructed by the method of column and row splitting, we have also examined several codes in a family of quasi-cyclic FG LDPC codes [9, 14] that perform well for our QKE protocol. Another possible task is to further enhance the QKE protocol. For example, the matrix $\boldsymbol{E_1}$ is not unique. If we have a way to search for an $\boldsymbol{E_1}$ having density as low as possible, then the block error rate of the code may not affect the bit error rate of the key by as much.

ACKNOWLEDGMENTS

T.A.B. and K.-C.H. would like to acknowledge the Center for High Performance Computing and Communications at the University of Southern California, who have provided computing resources. T.A.B. and K.-C.H. also thank Min-Hsiu Hsieh for helpful information and advice, and an anonymous referee for a useful suggestion. This work was supported by NSF Grant No. CCF-0830801.

[1] Z. Luo and I. Devetak, Phys. Rev. A **75**, 010303 (2007).
[2] P. W. Shor and J. Preskill, Phys. Rev. Lett. **85**, 441 (2000).
[3] T. A. Brun, I. Devetak, and M.-H. Hsieh, Science **314**, 436 (2006).
[4] Y. Kou, S. Lin, and M. Fossorier, IEEE Trans. Inf. Theory **47**, 2711 (2001).
[5] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, IEEE Trans. Inf. Theory **50**, 2315 (2004).
[6] T. Camara, H. Ollivier, and J.-P. Tillich, "Constructions and Performance of Classes of Quantum LDPC Codes," (2005), quant-ph/0502086.
[7] M. Hagiwara and H. Imai, Proceedings of ISIT 2007 (2007), quant-ph/0701020.
[8] D. Poulin and Y. Chung, Quantum Inf. Comput. **8**, 987 (2008).
[9] M.-H. Hsieh, T. A. Brun, and I. Devetak, Phys. Rev. A **79**, 032340 (2009).
[10] M.-H. Hsieh, W.-T. Yen, and L.-Y. Hsu, IEEE Trans. Inf. Theory **57**, 1761 (2011).
[11] M. Ohata and K. Matsuura, "Constructing CSS Codes with LDPC Codes for the BB84 Quantum Key Distribution Protocol," (2007), quant-ph/0702184.
[12] D. Elkouss, A. Leverrier, R. Alléaume, and J. J. Boutros, Proceedings of ISIT 2009 (2009), 0901.2140.
[13] D. J. C. MacKay, IEEE Trans. Inf. Theory **45**, 399 (1999).
[14] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, IEEE Trans. Commun. **52**, 1038 (2004).