Privacy-Preserving Data Storage in Cloud Using Array BP-XOR Codes

Yongge Wang, *Member, IEEE*
KINDI Center for Computing Research, Qatar University, Qatar
and
Dept. SIS, UNC Charlotte, NC 28223, USA
Email: yongge.wang@uncc.edu

***Abstract*—LDPC codes, LT codes, and digital fountain techniques have received significant attention from both academics and industry in the past few years. By employing the underlying ideas of efficient Belief Propagation (BP) decoding process in LDPC and LT codes, this paper designs the BP-XOR codes and use them to design three classes of secret sharing schemes called BP-XOR secret sharing schemes, pseudo-BP-XOR secret sharing schemes, and LDPC secret sharing schemes. By establishing the equivalence between the edge-colored graph model and degree-two BP-XOR secret sharing schemes, we are able to design novel perfect and ideal 2-out-of-n BP-XOR secret sharing schemes. By employing techniques from array code design, we are also able to design other (n, k) threshold LDPC secret sharing schemes. In the efficient (pseudo) BP-XOR/LDPC secret sharing schemes that we will construct, only linear number of XOR (exclusive-or) operations on binary strings are required for both secret distribution phase and secret reconstruction phase. For a comparison, we should note that Shamir secret sharing schemes require $O(n \log n)$ field operations for the secret distribution phase and $O(n^2)$ field operations for the secret reconstruction phase. Furthermore, our schemes achieve the optimal update complexity for secret sharing schemes. By update complexity for a secret sharing scheme, we mean the average number of bits in the participant's shares that needs to be revised when certain bit of the master secret is changed. The extremely efficient secret sharing schemes discussed in this paper could be used for massive data storage in cloud environments achieving privacy and reliability without employing encryption techniques.**

***Index Terms*—Error correcting codes, edge colored graphs, perfect one factorization of complete graphs, secret sharing schemes, data search over encrypted cipher texts.**

I. Introduction

The concept of threshold secret sharing schemes is one of the most important cryptographic primitives that have been used in many areas of cryptographic applications. Since the concept of secret sharing schemes was introduced by Blakley [5] and Shamir [30], there have been considerable efforts on the study of the bounds of share sizes, information rate, the number of participants for perfect and ideal threshold schemes, and on efficient secret sharing schemes. By an ideal threshold scheme, we mean a secret sharing scheme for which the size of the shares is the same as the size of the secret.

In a simple secret sharing scheme, we have n participants. A secret s is encoded into n shares and each participant will receive one share. Any $k \leq n$ participants can come together and reconstruct the secret s though no $k-1$ participants could learn any information of the secret. These schemes are known as (n, k) threshold schemes.

One of the most important problems in secret sharing schemes is to make both secret distribution phase and secret reconstruction phase efficient. For the Reed-Solomon codes based Shamir secret sharing scheme, it takes $O(n \log n)$ field operations for the secret distribution phase and $O(n^2)$ field operations for the secret reconstruction phase. It should be noted that in theory, Reed-Solomon codes could be decoded in $O(n \log^2 n \log \log n)$ field operations (see, e.g., [25]). However, for small numbers of n, the quadratic time algorithms are much faster than the fast theoretical algorithms.

Low Density Parity Checking (LDPC) codes were invented by Robert Gallager [16] in his PhD thesis. After being invented, they were largely forgotten and have been reinvented multiple times for the next 30 years. In the last decade, LDPC codes found their ways to many applications such as satellite transmission of digital television, 10GBase-T Ethernet, and WiFi 802.11n standards. Luby transform codes (LT codes) [22] are erasure correcting codes and are instances of LDPC codes. LT codes belong to the family of rateless digital fountain codes which can in principle produce an infinite number of encoding symbols.

By employing the underlying ideas of efficient Belief Propagation (BP) decoding process in [22], [23], Wang [35] introduced the BP-XOR codes. Based on the array coded in [35], this paper introduces three classes of secret sharing schemes called BP-XOR secret sharing schemes, pseudo-BP-XOR secret sharing schemes, and LDPC secret sharing schemes. The BP-XOR secret sharing schemes can be considered as LT code based secret sharing schemes. Edge-colored graph models were introduced by Wang and Desmedt in [36] to model homogeneous faults in networks. We will show the equivalence between the edge-colored graph model and degree-one-and-two encoding symbol based BP-XOR secret sharing schemes. Using this equivalence result, we are able to design perfect and ideal 2-out-of-n BP-XOR secret sharing schemes. We will also present several designs for general (n, k) LDPC secret sharing schemes. In the (pseudo) BP-XOR/LDPC secret sharing schemes that we will construct, only XOR (exclusive-or) operations on binary strings are required and both the secret distribution phase and secret reconstruction phase could be done in linear number of XOR operations on bits (relative to the secret size).

One of the other important problems in secret sharing schemes is to design efficient homomorphic sharing schemes which were first introduced by Benaloh [3]. Desmedt and Frankel [13] proposed a black-box secret sharing scheme for which the distribution matrix and the reconstruction vectors are defined over the integer ring Z and are designed independently of the group G from which the secret and the shares are sampled. Cramer and Fehr [12] then continued the study by showing that the optimal lower bound $O(\log_2 n)$ for the expansion factor could be achieved in general Abelian groups. Secret sharing schemes, for which the secret distribution and reconstruction phases are based on XOR operations on binary strings, could be converted to secret sharing schemes on several black-box Abelian groups on binary strings. Thus our results could be adapted to certain black-box based secret sharing schemes.

In this paper, we will introduce the concept of *update complexity* of secret sharing schemes. The update complexity of a secret sharing scheme is defined as the average number of bits of the shares affected by a change of a single master secret bit. For an (n, k) threshold secret sharing scheme, it is straightforward that the lower bound for the update complexity is $n - k + 1$. In this paper, we will show that our schemes will achieve this lower bound. It should be noted that traditionally the efficiency of secret sharing schemes have been extensively studied to reduce the bounds of share sizes and reducing the computational cost in shares distribution and secret reconstruction. However, we are not aware of any research that addresses the update cost of secret sharing schemes. Since our scheme is exclusive-or based, we have achieved the optimal information theoretical bound of share sizes (i.e., our scheme is an ideal secret sharing scheme) and has the most efficient computational cost in shares distribution and secret reconstruction process. The update complexity for traditional secret sharing schemes may not be a serious concern since secret sharing schemes are normally used for sharing a short secret (e.g., 1000 bits). However, our scheme is based on exclusive-or operations only. Thus it is possible to use our secret sharing schemes to share large amount of data in distributed cloud environments. Thus it is important to consider the update complexity. For example, if a user stores 1GB data in cloud using our scheme and changes 1KB of the data, then she does not need to down-load the entire shares from all servers and reconstruct the shares (for traditional secret sharing schemes she has to do that). In our scheme, all she needs to do is to update at most 1KB of data on each cloud server.

Due to the expensive cost of field operations of secret sharing schemes such as Shamir schemes, secret sharing schemes have traditionally been used for the distribution of secret keys, which is then used to encrypt the actual data. Since our schemes are based on linear numbers of XOR operations and are extremely efficient, it is possible to share the massive data directly using secret sharing schemes. In another word, the data stored at different locations (e.g., cloud servers) are shares of the original data and only XOR operations on bits are needed for distributing massive data to the cloud servers and for reconstructing the original data from shares stored at different cloud servers. This may help to design efficient protocols for certain applications of computation over encrypted data.

It should be noted there have been several publications on XOR based secret sharing schemes. For example, Suga [32] introduced a "2-propagation bases set" notion as a bases set to show the existence of $(2; 2^m)$-threshold secret sharing schemes. Based on Fujii, et al [14] and Kurihara, et al's scheme [19], Suga [33] proposed some new constructions of $(2, n)$ secret sharing schemes based on exclusive-or operations. Lv, et al [24] proposed an improved exclusive-or secret sharing scheme based on Kurihara, et al's scheme [19].

The structure of this paper is as follows. In Section II, we briefly discuss the connection between MDS codes and secret sharing schemes and show how to covert an MDS code to a secret sharing scheme. Section III introduces the concepts of LDPC based secret sharing schemes such as BP-XOR secret sharing schemes, pseudo BP-XOR secret sharing schemes, and LDPC secret sharing schemes. Section IV introduces perfect and ideal $(n, 2)$ threshold BP-XOR secret sharing schemes. Section V discusses perfect and ideal (n, k) LDPC secret sharing schemes for $k = 3, 4$. We briefly discuss dual codes in Section VI and in Section VII we have some discussion on applying our techniques to reliable and private massive data storage in cloud. Section IX discusses constructions of general low update complexity secret sharing scheme design using low weight irreducible polynomials.

II. MDS codes and Secret Sharing Schemes

For an $[n, k, d]$ linear code, the Singleton bound claims that $d \leq n - k + 1$. An $[n, k, d]$ linear code is maximum distance separable (MDS) if $d = n - k + 1$ (see, e.g., [25]). A k out of n threshold sharing scheme is a perfect sharing scheme if for any secret s_0, any k participants can reconstruct the s_0 and any subset of $k - 1$ or less participants gain no information about s_0. Let $p_1, \cdots, p_n$ represent the shares distributed to the n participants and let s be a variable over the secret sampling space. We can restate these requirements as follows. For any set of k indices $\{i_1, i_2, \cdots, i_k\}$, we have

$$Prob(s = s_0 | p_{i_1}, p_{i_2}, \cdots, p_{i_k}) = 1 \tag{1}$$

and

$$Prob(s = s_0 | p_{i_1}, p_{i_2}, \cdots, p_{i_{k-1}}) = Prob(s = s_0). \tag{2}$$

It is well known [6] that each $[n, k, d]$ MDS code could be converted to a perfect and ideal (n, k) threshold sharing scheme. As an example, the polynomial defined in the following equation (3) is a generator polynomial $g(z)$ for the Reed-Solomon code over $F = GF(q^m)$ where γ is a primitive element of $GF(q^m)$.

$$\begin{aligned} g(z) &= (z-1)(z-\gamma)\cdots(z-\gamma^{n-k-1}) \\ &= g_0 + g_1 z + \cdots + g_{n-k} z^{n-k} \end{aligned} \tag{3}$$

For an information symbol polynomial $f(z) = f_0 + f_1 z + \cdots + f_{k-1} z^{k-1}$, the encoding symbol polynomial $c(z) = f(z)g(z)$.

For the above Reed-Solomon code, if we let the secret $s = f_0$ and distribute the remaining coefficients of $c(z)$ to the $n-1$

participants respectively, then we obtain a perfect and ideal $(n-1,k)$ threshold secret sharing scheme over $GF(q^m)$. By applying a technique from Karnin, Greene, and Hellman [18], this scheme could be extended to an (n,k) threshold secret sharing scheme if we give $c_n = f_0 + \cdots + f_{k-1}$ to the nth participant.

III. (Pseudo) BP-XOR Secret Sharing Schemes and LDPC Secret Sharing Schemes

Low density array codes have been studied extensively for burst error correction in communication systems and storage systems (see, e.g., [9], [10], [37]). Array codes are linear codes where information and parity data are placed in a two dimensional matrix array. In this section, we introduce several concepts of LDPC based secret sharing schemes. We first introduce the concepts of array BP-XOR codes which will then be used to define LDPC based secret sharing schemes.

Throughout the paper, we will use the message alphabet set $M = \{0,1\}$. For fixed numbers n, k, t, and b where $n \geq \max\{k,t\}$, let $v_1, \cdots, v_{bk}$ be variables taking values from M, which are called information symbols. A t-erasure tolerating $[n,k]$ array code is a $b \times n$ matrix $\mathbf{C} = [\alpha_{i,j}]_{1 \leq i \leq b, 1 \leq j \leq n}$ such that each encoding symbol $\alpha_{i,j} \in \{0,1\}$ is the exclusive-or (XOR) of one or more information symbols from $v_1, \cdots, v_{bk}$ and $v_1, \cdots, v_{bk}$ could be recovered from any $n-t$ columns of the matrix. For an encoding symbol $\alpha_{i,j} = v_{i_1} \oplus \cdots \oplus v_{i_\sigma}$, we call v_{i_j} $(1 \leq j \leq \sigma)$ a neighbor of $\alpha_{i,j}$ and call σ the degree of $\alpha_{i,j}$. A t-erasure tolerating $[n,k]$ $b \times n$ array code $\mathbf{C}$ is said to be maximum distance separable (MDS) if $k = n - t$.

The $[n,k]$ array code $\mathbf{C}$ over the alphabet M can be considered as a linear code over the extension alphabet M^b of length n or a linear code over the alphabet M of length bn. A $bt \times bn$ (respectively, $bk \times bn$) binary matrix is said to be a parity-check (respectively, generator) matrix of a $b \times n$ array code $\mathbf{C}$ if it is a parity-check (respectively, generator) matrix of $\mathbf{C}$ when $\mathbf{C}$ is considered as a length bn linear code over the alphabet M. For example, the matrix H (respectively G) is a parity-check (respectively, generator) matrix of the array code $\mathbf{C}$ if we have $H\mathbf{y}^T = 0$ (respectively, $\mathbf{y} = \mathbf{x}G$) where $\mathbf{y} = (\alpha_{1,1}, \cdots, \alpha_{b,1}, \cdots, \alpha_{1,n}, \cdots \alpha_{b,n})$, $\mathbf{x} = (v_1, \cdots, v_{bk})$, and the addition is defined as the XOR on bits. An array code $\mathbf{C}$ is called low density parity-check (LDPC) if its parity-check (or equivalently, the generator) matrix contains small number of nonzero entries. For an MDS array code, it is straightforward to show that each row of the parity-check (respectively, the generator) matrix must contain at least $n - t + 1$ (respectively, $t + 1$) nonzero entries (see [10] for a proof).

The Belief Propagation decoding process (also called message passing iterative decoding) for binary symmetric channels (BSC) is present in Gallager [16]. The BP decoding process for binary erasure channels (BEC) is described as follows:

> (Cf. [22], [23]) If there is at least one encoding symbol that has exactly one neighbor then the neighbor can be recovered immediately. The value of the recovered information symbol is XORed into any remaining encoding symbols that have this information symbol as a neighbor. The recovered information symbol is removed as a neighbor of these encoding symbols and the degree of each such encoding symbol is decreased by one to reflect this removal.

If we add the following additional step to each loop of the above BP decoding process, we obtain a new decoding process which we will call the pseudo-BP decoding process on the BEC:

> If all the neighbors of an encoding symbol σ_i is a subset of the neighbor set of another encoding symbol σ_j, then the value of σ_i is XORed into all such kind of encoding symbols σ_j.

By the above definition, the major difference between BP-decoding and pseudo-BP-decoding is that in BP decoding process, we are only allowed to add a single decoded symbol to an encoded symbol while in pseudo-BP decoding process, we are allowed to add an encoded symbol x to another encoded symbol y if x is a proper subset of y. From decoding complexity aspects, pseudo BP decoding process could be slightly slower than BP decoding process. However, it can be shown that there exist codes that could be pseudo-BP decoded but could not be BP decoded. This allows us to design more powerful codes using pseudo BP decoding process. However, we should also note that pseudo BP decoding process is not as powerful as the slower Gauss elimination process.

A t-erasure tolerating $[n,k]$ array code $\mathbf{C} = [\alpha_{i,j}]_{1 \leq i \leq b, 1 \leq j \leq n}$ is called an $[n,k]$ array BP-XOR code (respectively, array pseudo BP-XOR code) if all information symbols $v_1, \cdots, v_{bk}$ can be recovered from any $n-t$ columns of encoding symbols using the BP-decoding process on the BEC (respectively, the pseudo BP-decoding process on the BEC).

A t-erasure tolerating $b \times n$ array code $\mathcal{C}$ is said to be maximum distance separable (MDS) if we have $k = n - t$. The array code $\mathcal{C}$ over the alphabet $M = \{0,1\}^l$ can be considered as a linear code over the extension alphabet M^b of length n or a linear code over the alphabet M of length bn. A $bt \times bn$ (respectively, $bk \times bn$) binary matrix is said to be a parity-check (respectively, generator) matrix of a $b \times n$ array code $\mathcal{C}$ if it is a parity-check (respectively, generator) matrix of $\mathcal{C}$ when $\mathcal{C}$ is considered as a length bn linear code over the alphabet M. For example, the matrix H (respectively G) is a parity-check (respectively, generator) matrix of the array code $\mathcal{C}$ if we have $H\mathbf{y}^T = 0$ (respectively, $\mathbf{y} = \mathbf{x}G$) where $\mathbf{y} = (\sigma_{1,1}, \cdots, \sigma_{b,1}, \cdots, \sigma_{1,n}, \cdots \sigma_{b,n})$, $\mathbf{x} = (\alpha_1, \cdots, \alpha_{bk})$, and the addition of two strings in M is defined as the XOR on bits.

An array code $\mathcal{C}$ is called low density parity-check (LDPC) if its parity-check (or equivalently, the generator) matrix contains small number of nonzero entries. For an MDS array code, it is straightforward to show that each row of the parity-check (respectively, the generator) matrix must contain at least $n - r + 1$ (respectively, $r + 1$) nonzero entries (see [10] for a proof).

Now we are ready for our definition of LDPC secret sharing schemes.

Definition 3.1: An (n,k) threshold BP-XOR (respectively, pseudo BP-XOR or LDPC) secret sharing scheme $\mathcal{S}$ is an array BP-XOR (respectively, pseudo BP-XOR or LDPC) code $\mathcal{C} = [\sigma_{i,j}]_{1\le i\le b, 1\le j\le n+1}$ with the following conditions:

- the secret $s_0 = \sigma_{1,1}||\cdots||\sigma_{b,1}$ where $||$ denotes string concatenation.
- for $1 \le i \le n$, the ith participant receives the share $p_i = \sigma_{1,i+1}||\cdots||\sigma_{b,i+1}$
- the secret distribution process in the above two steps generates a perfect (n,k) threshold secret sharing scheme.

IV. Perfect and ideal $(n,2)$ threshold BP-XOR secret sharing schemes

In this section, we first show the equivalence of degree 2 array BP-XOR codes and edge-colored graphs. We then use these results to design array BP-XOR codes and finally we convert these array BP-XOR codes to efficient perfect and ideal $(n,2)$ threshold BP-XOR secret sharing schemes.

A. Edge-colored graphs

Wang and Desmedt [36] introduced the edge-colored graph model for modeling homogeneous faults in networks.

Definition 4.1: (Wang and Desmedt [36]) An edge-colored graph is a tuple $G(V,E,C,f)$, with V the node set, E the edge set, C the color set, and f a map from E onto C. The structure

$$\mathcal{Z}_{C,t} = \{Z : Z \subseteq E \text{ and } |f(Z)| \le t\}. \tag{4}$$

is called a *t-color* adversary structure. Let $A, B \in V$ be distinct nodes of G. A, B are called *$(t+1)$-color connected* for $t \ge 1$ if for any color set $C_t \subseteq C$ of size t, there is a path p from A to B in G such that the edges on p do not contain any color in C_t. An edge-colored graph G is *$(t+1)$-color connected* if and only if for any two nodes A and B in G, they are $(t+1)$-color connected.

Fig. 1. 3-color connected edge-colored graph $G_{4,2}$

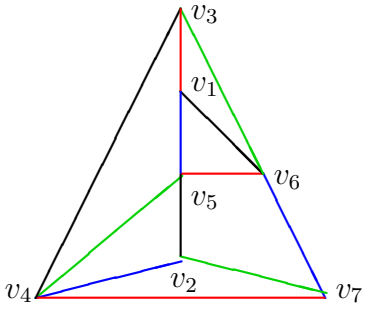

TABLE I
Table representation of edge-colored graph $G_{4,2}$

$\langle v_1, v_6\rangle$	$\langle v_2, v_7\rangle$	$\langle v_3, v_1\rangle$	$\langle v_4, v_2\rangle$
$\langle v_2, v_5\rangle$	$\langle v_3, v_6\rangle$	$\langle v_4, v_7\rangle$	$\langle v_5, v_1\rangle$
$\langle v_3, v_4\rangle$	$\langle v_4, v_5\rangle$	$\langle v_5, v_6\rangle$	$\langle v_6, v_7\rangle$

As an example, Figure 1 shows a 3-color connected graph $G_{4,2}$ with 7 nodes, 12 edges, and 4 colors. The edge-colored graphs $G_{4,2}$ can also be represented by the Table I where the edges with the same color are put in the same column.

Though Wang and Desmedt [36] presented a few simple constructions of edge-colored graphs with certain color connectivity, their results are not sufficient for our study of general array BP-XOR code design. In the following, we present a general construction of $(t+1)$-color connected edge-colored graphs using perfect one-factorizations of complete graphs. We use $K_n = (V,E)$ to denote the complete graph with n nodes. For an even n, a one-factor of K_n is a spanning 1-regular subgraph (or a perfect matching) of K_n. A one-factorization of K_n (n is even) is a set of one-factors that partition the set of edges E. A one-factorization is called perfect (or P1F) if the union of every two distinct one-factors is a Hamiltonian circuit. It is known (see, e.g., [26]) that perfect one-factorizations for K_{p+1}, K_{2p}, and certain K_{2n} do exist, where p is a prime number. It is conjectured that P1F exist for all K_{2n}. In the following Example 4.2, we show the typical constructions of perfect one factorization for K_{p+1} and K_{2p}.

Example 4.2:

- **P1F for K_{p+1}:** For an integer a, let $\langle a\rangle_p$ denote the integer $b \in \{0,\cdots,p-1\}$ such that $b \equiv a \bmod p$. Let $V = \{v_0, v_1, \cdots, v_p\}$ and

$$F_i = \{\langle v_i, v_p\rangle\} \cup \{\langle v_{\langle j_1+i\rangle_p}, v_{\langle j_2+i\rangle_p}\rangle : \langle j_1+j_2\rangle_p = 0 \text{ and } 0 \le j_1 \ne j_2 < p\} \tag{5}$$

  for $i = 0,\cdots,p-1$. Then $F_0, F_1, \cdots, F_{p-1}$ is a perfect one factorization of K_{p+1}.
- **P1F for K_{2p}:** Let $V = \{v_0,\cdots,v_{2p-1}\}$. For even i, let

$$F_i = \{\langle v_{j_1}, v_{j_2}\rangle : j_1 + j_2 = i \bmod 2p\} \cup \{\langle v_{\frac{i}{2}}, v_{\frac{i}{2}+p}\rangle\}, \tag{6}$$

  and for odd $i \ne p$, let

$$F_i = \{\langle v_{j_1}, v_{j_2}\rangle : j_1 \text{ is odd}, j_1 - j_2 = i \bmod 2p\}. \tag{7}$$

  Then $F_0, F_1, \cdots, F_{p-1}, F_{p+1}, \cdots, F_{2p-2}$ is a perfect one factorization of K_{2p}. **Another P1F for K_{2p} by Anderson [1]:** $V = \{w_0, w_1, \cdots, w_{p-1}, w_0^*, w_1^*, \cdots, w_{p-1}^*\}$. For $0 \le s \le p-1$, let

$$\begin{aligned} OG_s = \; & \{(w_i, w_j) : i + j = s, i \ne j \bmod p\} \cup \\ & \{(w_i^*, w_j^*) : i + j = p - 2 - s, i \ne j \bmod p\} \cup \\ & \{(w_{s/2}, w^*_{(p-2-s)/2})\} \end{aligned}$$

  where $1/2$ means $2^{-1} \bmod p$. For $0 \le s \le p-2$,

$$IG_s = \{(w_i, w_j^*) : i + j = s \bmod p\} \tag{8}$$

  Then $OG_0, OG_1, \cdots, OG_{p-1}, IG_0, IG_1, \cdots, IG_{p-2}$ is a perfect factorization of K_{2p}.

Theorem 4.3: Let n be an odd number such that there is a perfect one-factorization $F_1,\cdots,F_n$ for K_{n+1}. For each $t \le n-2$, there exists a $(t+1)$-color connected edge-colored graph G with n nodes, $(t+2)(n-1)/2$ edges, and $t+2$ colors.

Proof. Let $v_1,\ldots,v_{n+1}$ be a list of nodes for K_{n+1} and $V = \{v_1,\cdots,v_n\}$. Let $F_i' = F_i \setminus \{\langle v_{n+1}, v_j\rangle : j = 1,\cdots,n\}$, $E = F_1' \cup \cdots \cup F_{t+2}'$, and color all edges in F_i' with the color c_i for $i \le t+2$. Then it is straightforward to check that the edge-colored graph (V,E) is $(t+1)$-color connected, $|V| = n$, and $|E| = (t+2)(n-1)/2$. □

Remarks on Theorem 4.3: Since only node connectivity instead of Hamiltonian circuit is required for $(t+1)$-color connected graphs, we could use F_i' instead of F_i to construct edge-colored graphs. By using F_i' instead of F_i, we eliminate $t+2$ edges and one node in the resulting edge-colored graph. This helps us to keep the minimum cost for connectivity.

B. MDS array BP-XOR codes from edge-colored graphs

As an example, we first describe the array BP-XOR code corresponding to graph $G_{4,2}$ in Table I. Each edge in Table I is mapped to the XOR of the two adjacent nodes. Then choose a fixed node (e.g., v_7) and remove all occurrences of this node (e.g., v_7) to get the $[4,2]$ 3×4 array BP-XOR code in Table II.

TABLE II
BP-XOR CODE CORRESPONDING TO $G_{4,2}$

$v_1 \oplus v_6$	v_2	$v_3 \oplus v_1$	$v_4 \oplus v_2$
$v_2 \oplus v_5$	$v_3 \oplus v_6$	v_4	$v_5 \oplus v_1$
$v_3 \oplus v_4$	$v_4 \oplus v_5$	$v_5 \oplus v_6$	v_6

In the following, we give a general construction of array BP-XOR codes from edge-colored graphs. Let $v_1, v_2, \cdots, v_{bk}, v_{bk+1}$ be variables that take values from $M = \{0,1\}$. Let $G = (V, E, C, f)$ be a $(t+1)$-color connected edge-colored graph with $V = \{v_1, \cdots, v_{bk}, v_{bk+1}\}$, $|E| = m$, and $C = \{c_1, c_2, \cdots, c_n\}$. If we consider the nodes in $G = (V, E, C, f)$ as data block variables, edges as their parity check blocks of the adjacent nodes, and colors on the edges as labels for placing the parity checks into different columns of the array codes, then the following steps construct a $b\times n$ array BP-XOR codes, where $b = \max_{c\in C}\{|Z| : Z \subseteq E, f(Z) = c\}$.

1) For $1 \le i \le n$, let β_i be defined as
$$\begin{aligned}\beta_i = \{&v_{j_1} \oplus v_{j_2} : \langle v_{j_1}, v_{j_2}\rangle \in E,\\ &f(\langle v_{j_1}, v_{j_2}\rangle) = c_i, \text{ and } j_1, j_2 \neq bk+1\}\cup\\ &\{v_j : \langle v_j, v_{bk+1}\rangle \in E, f(\langle v_j, v_{bk+1}\rangle) = c_i\}\end{aligned} \quad (9)$$
2) If $|\beta_i|$ is smaller than b, duplicate elements in β_i to make it a b-element set.
3) The array BP-XOR code is specified by the $b\times n$ matrix $\mathbf{C}_G = (\beta_1^T, \cdots, \beta_n^T)$.

Next we show that the above array BP-XOR code $\mathbf{C}_G$ can tolerate t-erasure columns. Assume that the missing t columns of the code $\mathbf{C}_G$ correspond to the t-color set $C_t \subset C$ of the graph G. Since the graph G is $(t+1)$-color connected, for any node $v_{i_0} \in V$, we have a path $p = \langle v_{bk+1}, v_{i_1}, v_{i_2}, \cdots, v_{i_j}, v_{i_0}\rangle$ without using any colors in C_t. Thus v_{i_0} could be recovered by the following equation

$$v_{i_0} = v_{i_1} \oplus (v_{i_1} \oplus v_{i_2}) \oplus \cdots \oplus (v_{i_j} \oplus v_{i_0}) \quad (10)$$

where $v_{i_1}, v_{i_1} \oplus v_{i_2}, \cdots, v_{i_j} \oplus v_{i_0}$ are all available in the non-missing columns. In other words, the Belief Propagation decoding process could be used to recover the entire data blocks $v_1, \cdots, v_{bk}$ from the non-missing columns.

Theorem 4.4: Let n be an odd number such that there is a perfect one-factorization $F_1, \cdots, F_n$ for K_{n+1}. Then for $b = \frac{n-1}{2}$, there exists an $(n-2)$-erasure tolerating MDS $b\times n$ array BP-XOR code $\mathbf{C}_{b,n,2}$.

Proof. By Theorem 4.3, there exists a $(t+1)$-color connected edge-colored graph G with n nodes, $(t+2)(n-1)/2$ edges, and $t+2$ colors. By the above discussion, this $(t+1)$-color connected edge-colored graph G could be converted to an $(n-2)$-erasure tolerating MDS $b\times n$ array BP-XOR code $\mathbf{C}_{b,n,2}$. □

Corollary 4.5: For a given b, let $n \le 2b+1$. If $2b+1$ or $b+1$ is a prime number, then there exists an $(n-2)$-erasure tolerating MDS $b\times n$ array BP-XOR code $\mathbf{C}_{b,n,2}$.

C. Edge-colored graphs from array BP-XOR codes

In this section, we show that each array BP-XOR code could be converted to a corresponding edge-colored graph.

Theorem 4.6: Let $\mathbf{C}$ be an $b\times n$ array BP-XOR code with the following properties:

1) $\mathbf{C}$ is t-erasure tolerating;
2) $\mathbf{C}$ contains bk information symbols; and
3) $\mathbf{C}$ contains only degree one and two encoding symbols.

Then there exists a $(t+1)$-color connected edge-colored graph $G = (V, E, C, f)$ with $|V| = bk+1$, $|E| = bn$, and $|C| = n$.

Proof. Let $v_1, \cdots, v_{bk}$ be the information symbols of $\mathbf{C} = [\alpha_{i,j}]_{(i,j)\in[1,b]\times[1,n]}$ and $v_{i_1}, \cdots, v_{i_u}$ be a list of degree one encoding symbols in $\mathbf{C}$. Then the $(t+1)$-color connected edge-colored graph $G = (V, E, C, f)$ is defined by the following steps:

1) $V = \{v_1, \cdots, v_{bk}, v_{bk+1}\}$;
2) $E = \cup_{j\in[1,u]}\{\langle v_{bk+1}, v_{i_j}\rangle\} \cup \{\langle v_i, v_j\rangle : v_i \oplus v_j \in \mathbf{C}\}$;
3) $C = \{c_1, \cdots, c_n\}$;
4) Let $\alpha_{i,j} \in \mathbf{C}$. If $\alpha_{i,j} = v_{i'} \oplus v_{j'}$ then let $f(\langle v_{i'}, v_{j'}\rangle) = c_j$. Otherwise if $\alpha_{i,j} = v_{i'}$, let $f(\langle v_{bk+1}, v_{i'}\rangle) = c_j$.

Let C_t be a color set of size t and v_i and v_j be two nodes. Since the code $\mathbf{C}$ is t-erasure tolerating, both v_i and v_j could be recovered from encoding symbols not contained in the columns corresponding to the colors in C_t. Thus there exists a path p (respectively, q) connecting v_{bk+1} to v_i (respectively, to v_j) without using C_t-colored edges. It follows that $G = (V, E, C, f)$ is $(t+1)$-color connected. □

D. Perfect and ideal 2-out-of-n BP-XOR secret sharing schemes with minimum update complexity

By Corollary 4.5, there are MDS $b\times n$ array BP-XOR codes $\mathcal{C}_{b,n,2} = [\sigma_{i,j}]$ if $2b+1$ or $b+1$ is a prime number and $n \le 2b+1$. Following the discussion in Section II, we can convert $\mathcal{C}_{b,n,2}$ to a perfect and ideal $(n-1, 2)$ threshold BP-XOR secret sharing schemes as follows.

From our discussion in the previous section, the first column of $\mathcal{C}_{b,n,2}$ corresponds to a one-factor of an edge-colored graph. Without loss of generality, we may assume that the first column of $\mathcal{C}_{b,n,2}$ consists of $v_{i_1} \oplus v_{i_1'}$, $v_{i_2} \oplus v_{i_2'}$, $\cdots$, $v_{i_b} \oplus v_{i_b'}$. For a given secret $s = (s_1, \cdots, s_b)$ where $s_i \in \{0,1\}^l$, the dealer chooses random values for $v_{i_1'}, \cdots, v_{i_b'} \in \{0,1\}^l$ and sets $v_{i_1} = v_{i_1'} \oplus s_1$, $v_{i_2} = v_{i_2'} \oplus s_2$, $\cdots$, $v_{i_b} = v_{i_b'} \oplus s_b$. The dealer computes the code $\mathcal{C}_{b,n,2}$ using the P1F of the complete

graph K_{2b+2} and securely distributes the $(i+1)$th column of $\mathcal{C}_{b,n,2}$ to the participant p_i.

By the MDS property of $\mathcal{C}_{b,n,2}$, any two participants could use their shares to reconstruct the secret. Again by the MDS property of $\mathcal{C}_{b,n,2}$ and the discussion in Section II, the above constructed secret sharing scheme is a perfect and ideal 2-out-of-$(n-1)$ threshold secret sharing scheme.

It should be noted that the above constructed secret sharing schemes have the minimum update complexity. By the property that each column of $\mathcal{C}_{b,n,2}$ corresponds to a subset of a one-factor of the edge-colored graph together with zero or one node in the graph, each node of the edge-colored graph have at most one occurrence in each column of $\mathcal{C}_{b,n,2}$. In our above secret sharing scheme, the secret corresponds to one half of the nodes in the edge-colored graph. Thus if one bit of the master secret changes, we need to update at most one bit for the shares of each participant (some participant's shares do not need to be updated if her shares do not contain the corresponding node). Obviously, this is best one could achieve. In another word, our secret sharing schemes achieve the minimum update complexity.

E. An example of $(n,2)$ BP-XOR secret sharing schemes

Let n be the given number of participants for a desired perfect $(n,2)$ BP-XOR secret sharing scheme. The dealer finds the smallest p (or $2p$) such that $n \leq p$ (or $n \leq 2p-1$), where p is an odd prime. By Example 4.2 for K_{p+1}, we get the perfect one-factorization of K_{p+1} with node set $V = \{v_0, \cdots, v_p\}$ and the ith factor F_i as

$$\left\{\langle v_i, v_p\rangle; \langle v_{\langle 1+i\rangle_p}, v_{\langle p-1+i\rangle_p}\rangle; \cdots; \langle v_{\langle \frac{p-1}{2}+i\rangle_p}, v_{\langle \frac{p+1}{2}+i\rangle_p}\rangle\right\} \tag{11}$$

for $0 \leq i \leq p-1$. First we remove the edge $\langle v_i, v_p\rangle$ from F_i and the remaining edges in F_i are mapped to the XOR of the adjacent node variables. Then remove all occurrences of v_0 and we get the MDS $[n,2]$ $b \times p$ array BP-XOR code in Table III where $b = (p-1)/2$. Assume that the secret

TABLE III
$(p-1)/2 \times p$ BP-XOR CODE

$v_1 \oplus v_{p-1}$	$\cdots$	$v_{p-1} \oplus v_{p-3}$	v_{p-2}
$v_2 \oplus v_{p-2}$	$\cdots$	v_{p-4}	$v_1 \oplus v_{p-3}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$v_b \oplus v_{b+1}$	$\cdots$	$v_{b-2} \oplus v_{b-1}$	$v_{b-1} \oplus v_b$

$s = s_1 \cdots s_b$ where $s_i \in \{0,1\}^l$ (we can always pad the secret to a multiple of b-bits if it is not). The dealer chooses random values for $v_{b+1}, \cdots, v_{p-1}$ from $\{0,1\}^l$ and sets $v_1 = s_1 \oplus v_{p-1}, \cdots, v_b = s_b \oplus v_{b+1}$. For $i = 1, \cdots, n$, the ith participant receives the values in the $(i+1)$th column of Table III. It is straightforward to verify that the above scheme is a perfect and ideal $(n,2)$ secret sharing scheme.

V. PERFECT AND IDEAL (n,k) LDPC SECRET SHARING SCHEMES WITH MINIMUM UPDATE COMPLEXITY FOR $k = 3,4$

The results in the previous section show that for the design of perfect and ideal $(n,3)$ and $(n,4)$ LDPC secret sharing schemes, we need to use high degree encoding symbols. In this section, we show how to convert two array codes in the literature to lowest density perfect and ideal $(n,3)$ and $(n,4)$ LDPC secret sharing schemes.

For a prime number p, a class of low density $(p-1)/k \times (p-1)$ array codes were introduced in Blaum and Roth [10] as a generalization of the code in Zaitsev, Zinov'ev, and Semakov [15], yet the resulting codes are not necessarily MDS. Louidor and Roth [21] showed that when 2 is primitive in F_p then the resulting array codes are the lowest density MDS array codes for $k = 3,4$. It was conjectured that there are infinitely many primes p such that 2 is primitive in F_p. Though these codes are LDPC array codes, they are not array BP-XOR codes. Thus the secret sharing schemes that we will obtain from these codes are perfect and ideal LDPC secret sharing schemes. But these codes could be converted to perfect but non-ideal (pseudo) BP-XOR secret sharing schemes (the details will be presented in the full version of this paper).

For the construction of array codes in [21], let p be a prime such that 2 is primitive in F_p. In the finite field F_p, pick an element α of multiplicative order $k = 3,4$ and an element β of multiplicative order $p-1$. Let $C_{-1} = \{0\}$ and

$$C_0 = \{\alpha^0, \alpha^1, \cdots, \alpha^{k-1}\} \tag{12}$$

be the cyclic subgroup generated by α. For $1 \leq i < \frac{p-1}{k}$, let $C_i = \beta^i C_0$ be the coset of C_0. Then $C_{-1}, C_0, \cdots, C_{\frac{p-1}{k}-1}$ is a partition of $\{0,1,\cdots,p-1\}$. For $(i,j) \in [-1, \frac{p-1}{k}-1] \times [0, p-2]$, let

$$D'_{i,j} = \langle C_i + j\rangle_p \tag{13}$$

where $\langle C_i + j\rangle_p$ denotes the set that is obtained by adding j to the element of C_i modulo p.

It should be noted that exactly one of the sets $D'_{-1,j}$, $D_{0,j}$, $\cdots$, $D_{\frac{p-1}{k}-1,j}$ contains $p-1$. For each $j \in [0, p-2]$, let $D_{0,j}, D_{1,j}, \cdots, D_{\frac{p-1}{k}-1,j}$ be a list of the sets $D'_{i,j}$ such that $p-1 \notin D'_{i,j}$.

Define the $(p-1)/k \times (p-1)$ array code $\mathcal{C}_k = [\sigma_{i,j}]$ by letting $\sigma_{i,j}$ be the XOR of all elements in $D_{i,j}$. The authors in [21] showed that the array code $\mathcal{C}_k$ $(k = 3,4)$ is an MDS code tolerating $p-1-k$ erasure columns or $\lfloor \frac{p-k-1}{2} \rfloor$ columns of errors.

Using the techniques that we have used in Section IV, the above discussed LDPC MDS array codes $\mathcal{C}_k = [\sigma_{i,j}]_{i \in [1,(p-1)/k], j \in [1,p-1]}$ could be converted to perfect and ideal 3-out-of-$(p-2)$ and 4-out-of-$(p-2)$ LDPC secret sharing schemes by assigning the value of the secret s to the variables $\sigma_{1,1}, \cdots, \sigma_{(p-1)/k,1}$ in the first column in the same way as we have done in Section IV. Based on the design of the array codes $\mathcal{C}_k$, it is straightforward to verify that if we change any single bit in the master secret, then we only need to change at most one bit in the shares of each participant. Indeed, since the code $\mathcal{C}_k$, has the lowest density among all MDS codes of the same size [21], the corresponding secret sharing schemes have the least update complexity.

VI. Dual codes and perfect and ideal (n, k) LDPC secret sharing schemes for $k = n-1, n-2, n-3$

In the previous sections, we have described efficient perfect and ideal $(n, 2)$ BP-XOR secret sharing schemes and perfect and ideal (n, k) LDPC secret sharing schemes for $k = 3, 4$. It should be noted that the dual of these secret sharing schemes are also secret sharing schemes.

For the MDS $b \times n$ array BP-XOR codes $\mathcal{C}_{b,n,2}$ with $2b+1$ or $b+1$ being prime and $n \leq 2b+1$ in Section IV-D, the dual code $\mathcal{C}^D_{b,n,2}$ tolerates two erasure columns. If we use one column to define the secret, then it will tolerate one erasure column. Thus the array BP-XOR codes $\mathcal{C}^D_{b,n,2}$ will allow us to design $(n-2)$-out-of-$(n-1)$ perfect and ideal BP-XOR secret sharing schemes.

For the MDS LDPC $(p-1)/k \times (p-1)$ array codes with $k = 3, 4$ in Section V, the dual codes are MDS LDPC $(p-1)/k \times (p-1)$ array codes that could tolerating k erasure columns. If we use one column to define the secret, then it will tolerate $k-1$ erasure columns. Thus the dual codes could be used to design perfect and ideal $(p-k-1)$-out-of-$(p-2)$ lowest density LDPC secret sharing schemes for $k = 3, 4$. In another word, we can design perfect and ideal $(n-2)$-out-of-n (respectively, $(n-3)$-out-of-n) LDPC secret sharing schemes if $n+2$ is prime and 2 is primitive in F_{n+2}.

VII. Private and reliable cloud storage

Due to the advancement of cloud computing technologies, there has been an increased interest for individuals and business entities to move their data from traditional private data center to cloud servers. Indeed, even popular storage service providers such as Dropbox use third party cloud storage providers such as Amazon's Simple Storage Service (S3) for data storage. With the wide adoption of cloud computing and storage technologies, it is important to consider data privacy and reliability issues for cloud storage services.

In order to achieve reliability, data should be stored in multiple cloud storage servers with redundancy. If no privacy is required, then we can use the MDS BP-XOR codes (respectively, LDPC codes) constructed in Section IV-B (respectively, Section V) to store data within n cloud servers and the original data could be recovered from any 2 (respectively, 3 or 4) of the surviving servers. This will be sufficient for most cloud based applications.

If the user is also concerned with data privacy and does not trust a single cloud server, the user may use the perfect and ideal $(n, 2)$ threshold BP-XOR secret sharing schemes in Section IV or the perfect and ideal (n, k) threshold LDPC secret sharing schemes with $k = 3, 4$ in Section V to store the data in n servers. By using these secret sharing schemes, no single cloud server (respectively, 2 or 3 cloud servers jointly) could learn any information of the data. These schemes are very efficient since only linear number (in size of the data) of XOR operations are needed for data storage and recovery. This kind of efficiency is impossible with existing schemes such as Shamir secret sharing schemes.

Privacy preserving data update is simple in BP-XOR or LDPC secret sharing scheme based cloud storage. As an example, if the 2-out-of-n secret sharing scheme based on Table III is used, then the first column of $\mathcal{C}_{b,n,2}$ corresponds to the data file. In other words,

$$F = v_1 \oplus v_{p-1} || v_2 \oplus v_{p-2} || \cdots || v_b \oplus v_{b+1} \tag{14}$$

where $v_i \in \{0,1\}^l$. Now assume that the first bit of F is flipped. This is equivalent to flipping the first bit of v_{p-1}. Thus the data owner only needs to inform each server to flip one bit at certain location without leaking any other information.

In Section IV-D, we converted BP-XOR codes $\mathcal{C}_{b,n,2}$ to perfect and ideal 2-out-of-$(n-1)$ secret sharing schemes by letting the first column of $\mathcal{C}_{b,n,2}$ to represent the secret data. Indeed, we may also use a keyed pseudo random function $\mathcal{H}$ to generate the random bits in the secret distribution phase and obtain secret sharing schemes with some additional useful properties.

In the following, we use a prefix (wildcard) search example to show how the user could carry search in cloud environments without down-load the entire data to the local drive. A given data file F is divided into words (the searchable unit) of length l. In other words, $F = s_1 s_2 \cdots$ where $s_i \in M = \{0,1\}^l$. Without loss of generality, we may assume that $F = s_1 \cdots s_b$ (larger files could be divided into blocks of size b and processed separately).

In order to design a scheme that allows privacy preserving prefix search on the shares, we use a keyed pseudo random function $\mathcal{H}$ to generate the random bits during the secret distribution phase. In Table III, the first column of $\mathcal{C}_{b,n,2}$ corresponds to a one-factor of a complete graph: $v_1 \oplus v_{p-1}$, $v_2 \oplus v_{p-2}$, $\cdots$, $v_b \oplus v_{b+1}$. For the data file $F = s_1 \cdots s_b$ with identifier $\mathbf{id}$, the data owner sets

$$v_1 = \mathcal{H}(\mathbf{key}, \mathbf{id}, 1), \cdots, v_b = \mathcal{H}(\mathbf{key}, \mathbf{id}, b) \in \{0,1\}^l \tag{15}$$

where $\mathbf{key}$ is a secret that the data owner holds, and sets

$$v_{p-1} = v_1 \oplus s_1, v_{p-2} = v_2 \oplus s_2, \cdots, v_{b+1} = v_b \oplus s_b. \tag{16}$$

The data owner computes the code $\mathcal{C}_{b,n,2}$ and stores the $(i+1)$th column of $\mathcal{C}_{b,n,2}$ in the ith cloud server. Note that in this implementation, the entire data file F could either be recovered from any two cloud servers or from the secret **key** value and the entire data from one cloud server together with at most one additional block of data from a second cloud server, though any single server learns zero information about F (due to the perfect secret sharing property). This kind of scheme may have various applications for cloud data storage. For example, it may allow the application to build certain kind of secure computation over the encrypted data (that is, the shares at the storage servers).

Assume that the data owner wants to search whether the data file F contains a word with prefix s with $|s| \leq l$. For different cloud servers, the data owner needs to create different search queries. Assume that the query is submitted to the first server that stores: $v_2, v_3 \oplus v_{p-1}, \cdots, v_{b+1} \oplus v_{b+2}$. In order to check whether s_1 has prefix s, it is sufficient to check whether $v_1 \oplus v_{p-1}$ has prefix s. This is equivalent to show that the share $v_3 \oplus v_{p-1}$ in the server has prefix $v_3 \oplus v_1 \oplus s$ (here we reduce both v_3 and v_1 to the size $|s|$ by only using the first $|s|$ bits).

Since the values of v_1 and v_3 could be generated from the secret **key**, the data owner will be able to generate this query. It should be noted that neither the value of s nor the value of the original data file is leaked during this query.

It is straightforward to convert a privacy preserving prefix search scheme to an efficient privacy preserving keyword search scheme. This follows from the following observation: For a file F, one can put the keywords of F as the prefix of F and obtain $F' = s_1 \cdots s_b$ where $F = s_2 \cdots s_b$ and s_1 begins with the keywords of F. Thus it is sufficient to search whether s_1 contains the keyword.

Similarly, the keyed pseudo random function techniques could be applied to the 3-out-of-n (respectively, 4-out-of-n) secret sharing schemes in Section V so that the original data file could be recovered from either 3 surviving servers (respectively, 4 surviving servers) or from the secret key and 2 surviving servers (respectively, 3 surviving servers). Note that for these schemes, we do not need the additional data block from the 3rd server (respectively, the 4th server).

VIII. An example of perfect and ideal 3-out-11 LDPC secret sharing schemes

In this section, we show an example of a perfect and ideal 3-out-11 LDPC secret sharing scheme. Since 2 is primitive in F_{13}, we can choose $p = 13$, $\alpha = 3$, and $\beta = 2$. Then we have

$$\begin{array}{lll} C_{-1} = \{0\} & C_0 = \{1,3,9\} & C_1 = \{2,6,5\} \\ C_2 = \{4,12,10\} & C_3 = \{8,7,11\} & \end{array} \quad (17)$$

Following the process described in Section V, we have the 4×12 array code in Table IV. Note that for consistence with our other notations, we replaced 0 with 12 in Table IV.

For a 3-out-of-11 secret sharing scheme, let the master secret $s = s_1||s_2||s_3||s_4$ where $s_i \in \{0,1\}^l$. Choose random values for $v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}$ from $\{0,1\}^l$ and let $v_1 = s_1$, $v_2 = s_2 \oplus v_4 \oplus v_{10}$, $v_3 = s_3 \oplus v_6 \oplus v_7$, and $v_{12} = s_4 \oplus v_5 \oplus v_{11}$. Then we get the perfect and ideal LDPC 3-out-of-11 secret sharing scheme in Table V.

IX. Low density array codes based on low weight irreducible polynomials and secret sharing schemes with small update complexity

In previous sections, we used array BP-XOR codes to design secret sharing schemes that could be used in privacy preserving cloud data storage with optimal update complexity. For general k, we do not know whether such kind of (n,k) XOR secret sharing schemes exist. Alternatively, we may design non XOR based secret sharing schemes with small update complexity. One such potential approach is to use low weight irreducible polynomials.

For a binary polynomial $f(x)$, the weight of $f(x)$ is defined as the number of non-zero coefficients in $f(x)$. Low-weight irreducible polynomials over F_2 have been extensively studied for the efficient implementation of arithmetic operations in finite field F_{2^m} (see, e.g., [2], [17], [20], [28], [31], [34]), as the number of operations in the reduction of the product of two polynomials of degree $m-1$ modulo an irreducible of degree m and weight w is proportional to $(w-1)m$.

Though the general question of whether an irreducible polynomial over F_2 of weight at most 5 (or any other fixed odd weight w) exists for every value of n is an open one, the table of low-weight binary irreducible polynomials over F_2 in Seroussi [29] shows that for all $b \leq 10000$, there exists at least one irreducible trinomial $x^b + x^j + 1$ or one irreducible pentanomial $x^b + x^{j_1} + x^{j_2} + x^{j_3} + 1$ over F_2. If we represent a trinomial with (b, j) and a pentanomial with (b, j_1, j_2, j_3), the following is a list of irreducible polynomials for $b \leq 19$.

$(2,1)$	$(3,1)$	$(4,1)$	$(5,2)$	$(6,1)$
$(7,1)$	$(8,4,3,1)$	$(9,1)$	$(10,3)$	$(11,2)$
$(12,3)$	$(13,4,3,1)$	$(14,5)$	$(15,1)$	$(16,5,3,1)$
$(17,3)$	$(18,3)$	$(19,5,2,1)$		

For a polynomial $f(x) = x^b + f_{b-1}x^{b-1} + \cdots + f_0$ over F_2, let C_f be the companion matrix of $f(x)$ over F_2.

$$C_f = \begin{bmatrix} 0 & 0 & \cdots & 0 & f_0 \\ 1 & 0 & \cdots & 0 & f_1 \\ 0 & 1 & \cdots & 0 & f_2 \\ \vdots & \vdots & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & 1 & f_{b-1} \end{bmatrix}$$

By Calay-Hamilton theorem, we have $f(C_f) = 0$ and the order of C_f is $2^b - 1$.

For given n, k, $t = n-k$, and b with $k \leq 2^b - 1$, choose an irreducible polynomial $f(x)$ of degree b and define the block parity check matrix $H_{n,k,t,b}$ as

$$\begin{bmatrix} I & I & \cdots & I & I & 0 & 0 & \cdots & 0 \\ C_f & C_f^2 & \cdots & C_f^k & 0 & I & 0 & \cdots & 0 \\ C_f^2 & C_f^4 & \cdots & C_f^{2k} & 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_f^{t-1} & C_f^{2(t-1)} & \cdots & C_f^{k(t-1)} & 0 & 0 & 0 & \cdots & I \end{bmatrix}$$

where I is the $b \times b$ identity matrix. The above parity check matrix defines an extended Reed-Solomon code based $[n,k]$ MDS $b \times n$ array code. It should be noted that these array codes are generally not array BP-XOR codes. Thus Gaussian elimination techniques are used to recover the information symbols.

As an example, for $n = 6, k = 3$, and $b = 3$, one can use $f(x) = x^3 + x + 1$ as the irreducible polynomial. The corresponding parity check matrix $H_{6,3,3,3}$ is shown in equation (18)

$$H_{6,3,3,3} = \begin{bmatrix} 100 & 100 & 100 & 100 & 000 & 000 \\ 010 & 010 & 010 & 010 & 000 & 000 \\ 001 & 001 & 001 & 001 & 000 & 000 \\ \\ 001 & 010 & 101 & 000 & 100 & 000 \\ 101 & 011 & 111 & 000 & 010 & 000 \\ 010 & 101 & 011 & 000 & 001 & 000 \\ \\ 010 & 011 & 110 & 000 & 000 & 100 \\ 011 & 110 & 001 & 000 & 000 & 010 \\ 101 & 111 & 100 & 000 & 000 & 001 \end{bmatrix} \quad (18)$$

on the next page and the 3×6 array code is shown in Table

TABLE IV
4×12 ARRAY CODE (NOT ARRAY BP-XOR CODE)

v_1	v_2	v_3	v_4	v_5	v_6
$v_2 \oplus v_4 \oplus v_{10}$	$v_3 \oplus v_5 \oplus v_{11}$	$v_5 \oplus v_8 \oplus v_9$	$v_5 \oplus v_7 \oplus v_{12}$	$v_6 \oplus v_8 \oplus v_1$	$v_7 \oplus v_9 \oplus v_2$
$v_3 \oplus v_6 \oplus v_7$	$v_4 \oplus v_7 \oplus v_8$	$v_7 \oplus v_2 \oplus v_{12}$	$v_6 \oplus v_9 \oplus v_{10}$	$v_7 \oplus v_{10} \oplus v_{11}$	$v_{10} \oplus v_5 \oplus v_3$
$v_{12} \oplus v_5 \oplus v_{11}$	$v_9 \oplus v_{10} \oplus v_{12}$	$v_{10} \oplus v_{11} \oplus v_1$	$v_8 \oplus v_3 \oplus v_1$	$v_2 \oplus v_4 \oplus v_9$	$v_1 \oplus v_{12} \oplus v_4$
v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
$v_3 \oplus v_8 \oplus v_{10}$	$v_4 \oplus v_9 \oplus v_{11}$	$v_1 \oplus v_2 \oplus v_{11}$	$v_{11} \oplus v_{12} \oplus v_6$	$v_{12} \oplus v_3 \oplus v_4$	$v_1 \oplus v_3 \oplus v_9$
$v_4 \oplus v_6 \oplus v_{11}$	$v_{10} \oplus v_1 \oplus v_{12}$	$v_3 \oplus v_4 \oplus v_7$	$v_4 \oplus v_5 \oplus v_8$	$v_5 \oplus v_6 \oplus v_9$	$v_2 \oplus v_6 \oplus v_5$
$v_1 \oplus v_2 \oplus v_5$	$v_2 \oplus v_3 \oplus v_6$	$v_{12} \oplus v_8 \oplus v_6$	$v_1 \oplus v_7 \oplus v_9$	$v_2 \oplus v_8 \oplus v_{10}$	$v_8 \oplus v_7 \oplus v_{11}$

TABLE V
3-OUT-OF-11 LDPC SECRET SHARING SCHEME

secret part 1: s_1	$s_2 \oplus v_4 \oplus v_{10}$	$s_3 \oplus v_6 \oplus v_7$
secret part 2: s_2	$s_3 \oplus v_6 \oplus v_7 \oplus v_5 \oplus v_{11}$	$v_5 \oplus v_8 \oplus v_9$
secret part 3: s_3	$v_4 \oplus v_7 \oplus v_8$	$v_7 \oplus s_2 \oplus v_4 \oplus v_{10} \oplus s_4 \oplus v_5 \oplus v_{11}$
secret part 4: s_4	$v_9 \oplus v_{10} \oplus s_4 \oplus v_5 \oplus v_{11}$	$v_{10} \oplus v_{11} \oplus s_1$
v_4	v_5	v_6
$v_7 \oplus s_4 \oplus v_{11}$	$v_6 \oplus v_8 \oplus s_1$	$v_7 \oplus v_9 \oplus s_2 \oplus v_4 \oplus v_{10}$
$v_6 \oplus v_9 \oplus v_{10}$	$v_7 \oplus v_{10} \oplus v_{11}$	$v_{10} \oplus v_5 \oplus s_3 \oplus v_6 \oplus v_7$
$v_8 \oplus s_3 \oplus v_6 \oplus v_7 \oplus s_1$	$s_2 \oplus v_{10} \oplus v_9$	$s_1 \oplus s_4 \oplus v_5 \oplus v_{11} \oplus v_4$
v_7	v_8	v_9
$s_3 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{10}$	$v_4 \oplus v_9 \oplus v_{11}$	$s_1 \oplus s_2 \oplus v_4 \oplus v_{10} \oplus v_{11}$
$v_4 \oplus v_6 \oplus v_{11}$	$v_{10} \oplus s_1 \oplus s_4 \oplus v_5 \oplus v_{11}$	$s_3 \oplus v_6 \oplus v_4$
$s_1 \oplus s_2 \oplus v_4 \oplus v_{10} \oplus v_5$	$s_2 \oplus v_4 \oplus v_{10} \oplus s_3 \oplus v_7$	$s_4 \oplus v_5 \oplus v_{11} \oplus v_8 \oplus v_6$
v_{10}	v_{11}	$s_4 \oplus v_5 \oplus v_{11}$
$s_4 \oplus v_5 \oplus v_6$	$s_4 \oplus v_5 \oplus v_{11} \oplus s_3 \oplus v_6 \oplus v_7 \oplus v_4$	$s_1 \oplus s_3 \oplus v_6 \oplus v_7 \oplus v_9$
$v_4 \oplus v_5 \oplus v_8$	$v_5 \oplus v_6 \oplus v_9$	$s_2 \oplus v_4 \oplus v_{10} \oplus v_6 \oplus v_5$
$s_1 \oplus v_7 \oplus v_9$	$s_2 \oplus v_4 \oplus v_8$	$v_8 \oplus v_7 \oplus v_{11}$

TABLE VI
3×6 ARRAY $[6, 3, 4]$ CODE

v_1	v_4	v_7	$v_1 \oplus v_4 \oplus v_7$	$v_3 \oplus v_5 \oplus v_7 \oplus v_9$	$v_2 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8$
v_2	v_5	v_8	$v_2 \oplus v_5 \oplus v_8$	$v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9$	$v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_9$
v_3	v_6	v_9	$v_3 \oplus v_6 \oplus v_9$	$v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9$	$v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7$

VI.

Another potential approach to define Reed-Solomon codes using irreducible polynomials over F_2 is to use Cauchy matrices. Rabin [27] used Cauchy matrices to design information dispersal and retrieval protocols and Blomer et al [11] used Cauchy matrices to define XOR-based erasure-resilient coding schemes.

Let $t, k \leq 2^b - 1$. For an irreducible polynomial $f(x)$ over F_2 with C_f being the companion matrix. A $t \times k$ Cauchy block matrix $C = [c_{i,j}]_{i \in [1,t], j \in [1,k]}$ based on C_f is defined by letting $c_{i,j} = (C_f^i + C_f^{j+t})^{-1}$. That is, C equals to

$$\begin{bmatrix} (C_f + C_f^{t+1})^{-1} & (C_f + C_f^{t+2})^{-1} & \cdots & (C_f + C_f^{t+k})^{-1} \\ (C_f^2 + C_f^{t+1})^{-1} & (C_f^2 + C_f^{t+2})^{-1} & \cdots & (C_f^2 + C_f^{t+k})^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ (C_f^t + C_f^{t+1})^{-1} & (C_f^t + C_f^{t+2})^{-1} & \cdots & (C_f^t + C_f^{t+k})^{-1} \end{bmatrix}$$

By the fact that Cauchy determinant is non zero, every block square sub-matrix of a Cauchy block matrix is nonsingular (see, e.g., [27], [11]). Thus the parity check matrix $H^C_{n,k,t,b} = [C|I]$ defines an $[n, k]$ MDS $b \times n$ array code.

The authors in [11] used a different approach to design Cauchy matrix based Reed-Solomon codes over F_2. Specif-

ically, they first design Reed-Solomon codes using Cauchy matrices over F_{2^b}. Then they mapped each entry (an element of F_{2^b}) in the parity check matrix to a $b \times b$ matrix over F_2. The map is defined as follows. Assume that $f(x)$ is an irreducible polynomial of degree b over F_2. For each element $\mathbf{a} = (a_{1,1}, \cdots, a_{1,b}) \in F_{2^b}$, a $b \times b$ matrix $\tau(\mathbf{a}) = [a_{i,j}]$ is defined in [11] by letting $a_{i,1} + a_{i,2}x + \cdots + a_{i,b}x^{b-1} = x^{i-1}(a_{1,1} + a_{1,2}x + \cdots + a_{1,b}x^{b-1}) \mod f(x)$.

The above approach from [11] is equivalent to our companion matrix based approach by the following observation: For $\mathbf{a} = (0, 1, 0, \cdots, 0)$, $\tau(\mathbf{a})^T$ is the companion matrix of $f(x)$.

For $n = 6, k = 3, t = 3, b = 3$ and $f(x) = x^3 + x + 1$, $H_{6,3,3,3}$ is defined in equation (18) and $H^C_{6,3,3,3}$ is defined in equation (19).

$$H^C_{6,3,3,3} = \left[\begin{array}{cccccc} 111 & 001 & 010 & 100 & 000 & 000 \\ 011 & 101 & 011 & 010 & 000 & 000 \\ 101 & 010 & 101 & 001 & 000 & 000 \\ & & & & & \\ 110 & 011 & 100 & 000 & 100 & 000 \\ 001 & 110 & 010 & 000 & 010 & 000 \\ 100 & 111 & 001 & 000 & 001 & 000 \\ & & & & & \\ 001 & 111 & 101 & 000 & 000 & 100 \\ 101 & 100 & 111 & 000 & 000 & 010 \\ 010 & 110 & 011 & 000 & 000 & 001 \end{array}\right] \quad (19)$$

The numbers of non zero elements in the columns of $H_{6,3,3,3}$ and $H^C_{6,3,3,3}$ are listed in the following table. The table shows that Cauchy matrix based array codes have approximately 15% more non zero elements in the parity check matrix.

column	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**total**
$H_{6,3,3,3}$	3	4	5	4	6	5	5	4	5	41
$H^C_{6,3,3,3}$	5	4	6	6	6	5	4	5	6	47

In addition to the above comparison, we have run a sequence of experiments for different values of (n, k, t, b) and different low weight irreducible polynomials. The experimental results show that for any given low weight irreducible polynomial, the Cauchy matrix based parity check matrix $H^C_{n,k,t,b}$ consistently has 15% to 50% more non zero elements than $H_{n,k,t,b}$ (the 50% value comes from larger values of b). Thus for private preserving cloud storage systems, low weight irreducible polynomial based Vandermonde matrix may be a better choice for defining secret sharing schemes with low update complexity.

In order to further reduce the density of parity check matrices, it is useful to select a set of low weight irreducible polynomials with certain constraints, and the parity check matrix is constructed using the set of low weight irreducible polynomials instead of a single low weight irreducible polynomial.

Assume $f_1, \cdots, f_k$ is a list of low weight irreducible polynomials over F_2 of the same degree such that $C^j_{f_i}$ is relatively low density for $i \leq k$ and $j \leq n - k$. Let $C_i = P_i C_{f_i} P_i^{-1}$, where P_i is a permutation matrix. If $C_{i_1} + C_{i_2}$ is non-singular for all $i_1 \neq i_2$, then they could be used to construct the Vandermonde-like parity check matrices. Since $C^j_i = P_i C^j_{f_i} P_i^{-1}$, we can reduce the density of the final parity check matrix further.

Another approach that one may use is to investigate the use of low weight irreducible polynomials over F_{2^m}. In other words, one may use the companion matrices of low weight irreducible polynomials over F_{2^m} (where m divides b) to design array codes over F_{2^m}. These codes will then be converted to array codes over F_2 by replacing each element in the parity check matrix with the companion matrix corresponding to that element. The resulting array codes could then be used to design secret sharing schemes with low update complexity.

It is possible to construct Reed-Solomon and BCH codes over rings using non-irreducible polynomials (see, e.g., [4], [7], [8], [9]). Since the companion matrices C_f of certain non-irreducible polynomials have better properties for array code design, one may use these array codes to design secret sharing schemes with low update complexity.

For a given b, let $f_i(x) = x^b + x^i + 1$ and $C_{f_1}, \cdots, C_{f_{b-1}}$ be the companion matrices of these polynomials. As an example, for $b = 4$, $f_2(x) = x^4 + x^2 + 1$ and let C_{f_2} be the companion matrix of f_2.

$$\begin{array}{rcl} C_{f_1} & = & \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \\ C_{f_2} & = & \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right] \\ C_{f_3} & = & \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right] \end{array} \quad (20)$$

It is straightforward to check that $C^{15}_{f_1} = C^6_{f_2} = C^{15}_{f_3} = I_4$. Thus the order of C_{f_1}, C_{f_2}, and C_{f_3} are 15, 6, and 15 respectively (note that both f_1 and f_3 are irreducible polynomials). It is straightforward to check that $C^6_{f_2} = I_4$. Thus the order of C_{f_2} is 6.

By choosing appropriate permutation matrices Q_i, we may construct a list of low density matrices $C_i = Q_i C_{f_2} Q_i^{-1}$. If the list of matrices C_i has the properties that $C_{i_1} + C_{i_2}$ are non-singular for $i_1 \neq i_2$, then they may be used to design close to lowest density parity check matrices if each square sub-matrix of the Vandermonde matrix is non-singular. For example, for $p = 5$, the following matrices may be used

$$C_1 = \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right] \quad C_2 = \left[\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

and

$$C_3 = \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{array}\right] \quad C_4 = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{array}\right]$$

where $C_i = Q_i C_{f_2} Q_i^{-1}$ and Q_1, Q_2, Q_3, and Q_4 are the permutation matrices of the permutation $(1, 2, 3, 4)$,

$(3,1,4,2)$, $(2,4,1,3)$, and $(4,3,2,1)$ respectively. Furthermore, it should be noted that the numbers of non zero elements in $C^1_{f_2}, C^2_{f_2}, C^3_{f_2}, C^4_{f_2}, C^5_{f_2}$ are $5,6,6,6,5$ respectively, which are close to the lowest density that one could achieve.

X. Conclusion

In this paper, we used XOR secret sharing schemes to design privacy preserving data distribution schemes in cloud environments. This will allow computation over encrypted texts. However, we should also point out the limitation of this approach. For example, it will be generally hard to prevent collusion attacks. When deployed, the users should make the assumption that cloud servers will not collude which is hard to achieve in some cases. For example, we know that dropbox uses amazon servers for their data storage. In other words, dropbox and amazon cloud storage servers may be considered as one server in our applications.

Acknowledgment

I would like to thank anonymous referees for helpful comments on improving the presentation of this paper.

References

[1] BA Anderson. Symmetry groups of some perfect 1-factorizations of complete graphs. *Discrete Mathematics*, 18(3):227–234, 1977.

[2] M. Ben-Or. Probabilistic algorithms in finite fields. In *Proc. IEEE 22nd FOCS*, pages 394–398, 1981.

[3] J. Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proc. CRYPTO '86*, pages 251–260. Springer-Verlag, 1987.

[4] I.F. Blake. Codes over certain rings. *Information and Control*, 20(4):396–404, 1972.

[5] G. R. Blakley. Safeguarding cryptographic keys. *Managing Requirements Knowledge, International Workshop on*, 0:313, 1979.

[6] G. R. Blakley and G. A. Kabatianski. Ideal perfect threshold schemes and MDS codes. In *IEEE Conf. Proc., Int. Symp. Information Theory, ISIT' 95*, pages 488–488, 1995.

[7] M. Blaum. A family of mds array codes with minimal number of encoding operations. In *Information Theory, 2006 IEEE International Symposium on*, pages 2784–2788. IEEE, 2006.

[8] M. Blaum, J. Bruck, and E. Vardy. MDS array codes with independent parity symbols. *IEEE Trans. on Information Theory*, 42:529–542, 1996.

[9] M. Blaum and R. M. Roth. New array codes for multiple phased burst correction. *IEEE Trans. on Information Theory*, 39(1):66–77, 1993.

[10] M. Blaum and R. M. Roth. On lowest-density MDS codes. *IEEE Trans. on Information Theory*, 45:46–59, 1999.

[11] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman. An xor-based erasure-resilient coding scheme, 1995.

[12] R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In *CRYPTO*, pages 272–287, 2002.

[13] Y. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, 1994.

[14] Yoshihiro Fujii, M Tada, N Hosaka, K Tochikubo, and T Kato. A fast (2, n)-threshold scheme and its application. *Proc. CSS*, pages 631–636, 2005.

[15] N. V. Semakov G. V. Zaitsev, V. A. Zinov'ev. Minimum-check-density codes for correcting bytes of errors, erasures, or defects. *Problems Inform. Transmission*, 19(3):197–204, 1983.

[16] R. G. Gallager. *Low density Parity Check Codes*. MIT Press, 1963.

[17] S. Gao and D. Panario. Tests and constructions of irreducible polynomials over finite fields. In *Foundations of computational mathematics*, page 346. Springer Verlag, 1997.

[18] E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Trans. Information Theory*, 29(1):35 – 41, jan 1983.

[19] Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima, and Toshiaki Tanaka. A new (k, n)-threshold secret sharing scheme and its extension. In *Information Security*, pages 455–470. Springer, 2008.

[20] R. Lidl and H. Niederreiter. *Finite fields*, volume 20. Cambridge University Press, 1996.

[21] E. Louidor and R. M. Roth. Lowest density MDS codes over extension alphabets. *IEEE Trans. Inf. Theor.*, 52(7):3186–3197, 2006.

[22] M. Luby. LT codes. In *Proc. FOCS*, pages 271–280, 2002.

[23] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theor.*, 47:569–584, 2001.

[24] Chunli Lv, Xiaoqi Jia, Lijun Tian, Jiwu Jing, and Mingli Sun. Efficient ideal threshold secret sharing schemes based on exclusive-or operations. In *Network and System Security (NSS), 2010 4th International Conference on*, pages 136–143. IEEE, 2010.

[25] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. NH Pub. Company, 1978.

[26] R. Mendelsohn and A. Rosa. One-factorizations of the complete graph – a survey. *Journal of Graph Teory*, 9(1):43–65, 1985.

[27] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, April 1989.

[28] M.O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Computing*, 9(2):273–280, 1980.

[29] G. Seroussi. Table of low-weight binary irreducible polynomials. *HP LABORATORIES TECHNICAL REPORT HPL*, 1998.

[30] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[31] V. Shoup. Fast construction of irreducible polynomials over finite fields. In *Proc. 4th annual ACM-SIAM Symposium on Discrete algorithms*, pages 484–492, 1993.

[32] Yuji Suga. A fast (2, 2^ m)-threshold secret sharing scheme using m linearly independent binary vectors. In *Network-Based Information Systems (NBiS), 2013 16th International Conference on*, pages 539–544. IEEE, 2013.

[33] Yuji Suga. New constructions of (2, n)-threshold secret sharing schemes using exclusive-or operations. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 837–842. IEEE, 2013.

[34] J. Von Zur Gathen and V. Shoup. Computing frobenius maps and factoring polynomials. *Computational complexity*, 2(3):187–224, 1992.

[35] Yongge Wang. Array BP-XOR codes for reliable cloud storage systems. In *Proc IEEE ISIT 2013*, pages 326–330. IEEE Press, 2013.

[36] Yongge Wang and Yvo Desmedt. Edge-colored graphs with applications to homogeneous faults. *Inf. Process. Lett.*, 111(13):634–641, 2011.

[37] L. Xu, V. Bohossian, J. Bruck, and D. Wagner. Low density mds codes and factors of complete graphs. *IEEE Trans. Inf. Theor.*, 45:1817–1826, 1998.

Yongge Wang Yongge Wang received his PhD degree from the University of Heidelberg of Germany. Since then, Dr. Wang has worked in the industry for a few years until he joined UNC Charlotte in 2002. In particular, Dr. Wang has worked in Certicom (now a division of RIM) as a cryptographic mathematician specializing in efficient cryptographic techniques for wireless communications. Dr. Wang has been actively participated in and contributed to standard bodies such as IETF, W3C XML Security protocols, IEEE 1363 standardization groups for cryptographic techniques, ANSI X9 group for the financial services industry standards, and ANSI T11 groups for SAN network security standards. Dr. Wang is the inventor of Secure Remote Password authentication protocol SRP5 which is an IEEE 1363.2 standard and the inventor of identity based key agreement protocol WANG-KE which is an IEEE 1363.3 standard. Dr. Wang has also worked with Cisco researchers and American Gas Association researchers to design security protocols for the SCADA industry. Dr. Wang has published extensively on research topics including computational complexity, algorithmic information theory, randomness and pseudorandomness, critical infrastructure protection, perfectly secure message transmission, cryptography and secure authenticated communications, and statistical testing.