Computational Aspects of Ideal (t, n)-Threshold Scheme of Chen, Laing, and Martin

Mayur Punekar, Qutaibah Malluhi
KINDI Center for Computing Research,
Qatar University, Qatar.
Email: mayur.punekar@ieee.org,
qmalluhi@qu.edu.qa

Yvo Desmedt
The University of Texas at Dallas,
Richardson, TX, USA.
Email: yvo.desmedt@utdallas.edu

Yongge Wang
Dept. SIS, UNC Charlotte,
NC, USA.
Email: yongge.wang@uncc.edu

***Abstract*—In CANS 2016, Chen, Laing, and Martin proposed an ideal (t, n)-threshold secret sharing scheme (the *CLM scheme*) based on random linear code. However, in this paper we show that this scheme is similar to the one proposed by Karnin, Greene, and Hellman in 1983 (the *KGH scheme*). Further, the authors did not analyzed memory or XOR operations required to either store or calculate an inverse matrix needed for recovering the secret. In this paper, we analyze computational aspects of the CLM scheme and discuss various methods through which the inverse matrix required during the secret recovery can be obtained. Our analysis shows that for $n \leq 30$ all the required inverse matrices can be stored in memory whereas for $30 \leq n < 9000$ calculating the inverse as and when required is more appropriate. However, the CLM scheme becomes impractical for $n > 9000$. Another method which we discuss to recover the secret is to obtain only the first column of the inverse matrix using Lagrange's interpolation however, as we show, this method can be used with the KGH scheme only. Some potential application of the CLM and KGH schemes are also discussed.**

I. Introduction

Secret sharing refers to procedures in which a secret is split and shared among a group of participants or players. The secret can be reconstructed only when sufficient number of participants combine their shares together. However, individual shares gives no information about the secret. A type of secret sharing scheme, known as (t, n)-threshold secret sharing scheme can be used to distribute secret k to n participants in such a way that any t participants can uniquely recover the secret and at the same time any set of $t-1$ participants get no information about the secret. A secret sharing scheme for which the ratio of the size of the secret to the size of the largest share is 1 is called *ideal*.

Shamir and Blakley independently introduced (t, n)-threshold schemes in 1979 [1][2]. Blakley's method is based on linear projective geometry where each share specifies a hyperplane and the secret k is the unique point of intersection of the n hyperplanes. Shamir's scheme relies on a polynomial of degree $t-1$ to generate n shares and use Lagrange's interpolation method to recover the secret from t participants. However, Lagrange's interpolation is computationally intensive due to which efforts have been made to obtain schemes that have same properties as Shamir's scheme but can be implemented using XOR operations only.

First such XOR based (t, n)-threshold scheme was proposed by Kurihara *et al.* in [5] which was a generalization of their earlier work on $(3, n)$-threshold scheme [6]. Some other XOR based scheme were also proposed by Lv *et al.* [7][8], Wang *et al.* [9], and Chen *et al.* [10].

In CANS 2016, Chen, Laing, and Martin [3] proposed a (t, n)-threshold secret sharing scheme which is based on a patent application by HP [10]. The scheme is defined over $GF(2^\lambda)$ which can be generalized to any Galois field. It uses random linear code and requires only XOR and shift operations for distribution and recovery of the secret. The proof for security of the scheme is also given and it has been proven that the scheme is ideal.

However, we observe that the scheme by Chen *et al.* (the *CLM scheme*) is very similar to the one proposed earlier by Karnin, Grenne, and Hellman [4] in 1983 (the *KGH scheme*). Further, the CLM scheme relies on decoding a random linear code to recover the secret. The decoding method proposed by the authors inverts a $t \times t$ matrix G' which is derived by selecting t columns of the $t \times n$ generator matrix G used during distribution. Hence, the decoder either needs to calculate this inverse on the fly or have to store all possible such inverse matrices in memory. However, this important computational aspect of the CLM scheme has not been discussed by the authors in [3]. Their complexity analysis and comparison with other secret sharing schemes in [3, Fig. 2] neither discuss the memory required to store all inverse matrices nor the number of XOR operations needed to obtain such an inverse.

In this paper, we analyse the CLM scheme and focus on the computational aspects of it. First, four issues related to the CLM scheme are discussed, namely, similarity of the CLM with the KGH scheme, computational aspects related to the inverse of G', XOR operations required for vector and matrix multiplication and inability of the CLM scheme to detect and correct erroneous shares due to lack of efficient decoding algorithm. Further, we suggest and discuss three ways in which a inverse matrix can be obtained which is required in the CLM scheme: 1) store all possible inverse matrices, 2) calculate the inverse on the fly, 3) calculate only the first column of the inverse matrix. We also give estimate of the memory required to store all inverse matrices and the runtime required to invert a matrix for different n. Some potential application of the CLM scheme are also discussed.

The rest of the paper is structured as follows. In Sec. II we discuss Shamir's, the CLM and the KGH schemes. The CLM scheme is analyzed in detail in Sec. III. Different approaches

for inverse of G' are prosed and discussed in Sec. IV. Sec. V discuss some potential applications of the CLM and KGH schemes. We conclude the paper in Sec. VI.

II. Background

A. Shamir's Scheme

In Sharmir's (t, n)-threshold secret sharing scheme the secrets and shares are the elements of finite field $GF(q)$ for some prime-power $q > n$. n distinct non-zero elements $\alpha_1, \alpha_2, \ldots, \alpha_n \in GF(q)$ are selected which are known to all parties. Suppose a secret $k \in GF(q)$ needs to be shared among n parties. Then, $t-1$ random elements $a_1, \ldots, a_{t-1} \in GF(q)$ are chosen independently with uniform distribution. The secret along with the random numbers define a polynomial

$$P(x) = k + \sum_{i=1}^{t} a_i x^i \tag{1}$$

The share of party $v_j = P(\alpha_j)$ where $P(x)$ is evaluated using the arithmetic of $GF(q)$.

The Shamir's scheme's correctness and privacy properties can be proved using the Lagrange's interpolation theorem which states that : for every field $\mathbb{F}$, every t distinct values $x_1, \ldots, x_t$, and any t values $y_1, \ldots, y_t$, there exists a unique polynomial Q of degree at most $t-1$ over $\mathbb{F}$ such that $Q(x_j) = y_j$ for $1 \leq j \leq t$ [17].

The secret k can be recovered from t shares $v_{i_1}, \ldots, v_{i_t}$ using the following

$$Q(x) = \sum_{l=1}^{t} v_{i_l} \prod_{1 \leq j \leq t, j \leq l} \frac{\alpha_{i_j} - x}{\alpha_{i_j} - \alpha_{i_l}} \tag{2}$$

where the secret is given by $k = P(0) = Q(0)$.

B. Karnin, Greene, and Hellman Scheme

Karnin, Greene, and Hellman proposed ideal secret sharing scheme (***KGH scheme***) in [4]. The KGH scheme uses a vector u of length t whose first element is same as the secret $k \in GF(q^\lambda)$ and rest of the $t-1$ entries are generated randomly from $GF(q^\lambda)$ where q is a prime. A $t \times n+1$ systematic generator matrix G for some linear code whose entries are from $GF(q^\lambda)$ is used for encoding the vector u. The selected linear code must be MDS, i.e., any $t \times t$ submatrix derived from G by selecting any t columns must be invertible. The following relationship exit between the vector u and shares v [4, (13)],

$$v = uG \tag{3}$$

such that $v_0 = u_1 = k$ which is secret to be protected and $v_1, \ldots, v_{n+1}$ are the n shares to be distributed. It has been shown by the authors that the Shamir's scheme [1] is a special case of the proposed method.

The secret can be recovered when t shares are available for which the t linear equation over $GF(2^\lambda)$ with t unknown have to be solved.

C. Chen, Laing, and Martin Scheme

We now describe ideal (t, n)-threshold scheme proposed by Karnin, Greene, and Hellman (CLM scheme) in [3]. This scheme uses two algorithms referred to as $Share$ and $Recover$.

The $Share(k)$ algorithm [3, Fig. 1] takes a secret $k \in \{0,1\}^\lambda$ and parse it into t words where each word consists of $\lceil \frac{\lambda}{t} \rceil$ bits. If λ is not divisible by t, then k is padded with $(-\lambda) \mod t$ elements so that each word is an element in $F = GF(2^{\lceil \frac{\lambda}{t} \rceil})$ and $k \in F^t$. Then, $r_1, \ldots, r_{t-1} \in GF(2^\lambda)$ dummy keys are randomly generated and again parsed into t words of $\lceil \frac{\lambda}{t} \rceil$ bits. These dummy keys are XORed with secret k to produce k'. All dummy keys $r_1, \ldots, r_{t-1}$ and modified secret k' are dispersed using $Share^{\text{IDA}}$ algorithm [3, Sec. 2.3]. This algorithm multiplies the input vector with a systematic generator matrix G of an MDS linear code. The vector $\boldsymbol{K'}$ is obtained by applying $Share^{\text{IDA}}$ to K'. Similarly, $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_{t-1}$ are obtained from $r_1, \ldots, r_{t-1}$ using $Share^{\text{IDA}}$. A new $t \times n$ matrix M is created by using vectors K' and $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_{t-1}$ as its rows. Then the elements of row $i, 0 \leq i \leq t$ are are shifted to the left by i places to obtain a new matrix M'. The elements in column i are then concatenated and are used as shares.

The $Recover$ algorithm [3, Fig. 1] is used to recover the secret when at least t out of n shares are available. The algorithm retrieves the secret k using $Recover^{\text{IDA}}$ algorithm which is essentially an algorithm to decode the MDS linear code. However, before $Recover^{\text{IDA}}$ is used, the i-th share is shifted to the right by i places. $Recover^{\text{IDA}}$ algorithm creates a t-vector C' by using t pooled shares. Then, a $t \times t$ matrix G' is formed which consists of the t rows of generator matrix G corresponding to the t shares pooled. The matrix G' is then inverted and multiplied by the vector C' to obtain k' and $r_1, \ldots, r_{k-1}$ from which the secret k can be obtained using $k = k' \oplus r_1 \oplus \cdots \oplus r_{t-1}$.

III. Analysis of CLM scheme

In this section we analyse the CLM scheme and explain four issues which we observed.

First, we would like to point out that the CLM scheme is very similar to the KGH scheme explained in Sec. II-B. The only difference between these two schemes lies in the fact that, Chen *et al.* [3] add randomly generated words $r_1, \ldots r_{k-1}$ to the secret k before encoding it with $Share^{\text{IDA}}$ whereas this step is not used by Karnin *et al.* [4]. Also, it is not clear as to what advantage this additional step provides in terms of security or efficiency.

The second problem we observed is related to the decoder used in $Recover^{\text{IDA}}$ algorithm. The decoder needs to either compute the inverse of G' on the fly or store all possible $t \times t$ G' matrices in memory. In the first case, the XOR operations required to compute the inverse needs to be considered. For the second case, the memory required to store all possible matrices needs to be analysed. The comparison of the CLM scheme to other schemes as given in [3, Fig. 3] neither includes memory required to store all inverse matrices nor the number of XOR operations required to invert a matrix on the fly.

The third issue is with the vector and matrix multiplication used in $Recover^{\text{IDA}}$ algorithm. The number of XOR operations required for this multiplication has not been included in the comparison of CLM scheme with other schemes in [3, Fig. 3].

The fourth problem is related to the connections between the CLM and Shamir's scheme. It has been shown by Karnin *et al.* in [4] that if generator matrix G is chosen appropriately then KGH scheme is equivalent to the Shamir's scheme. However, in the CLM scheme the secret k is XORed with random words $r_1, \ldots, r_{k-1}$ to generate first word k' before encoding. This process though reduce the number of random words required in the CLM scheme has a downside that regardless of which generator matrix is chosen, the scheme does not have any relation to Shamir's scheme anymore. As shown by McEliece *et al.* in [11], Shamir's scheme corresponds to Reed-Solomon codes and hence Shamir's scheme has error detection and correction capability when more than t shares are available and Reed-Solomon decoding algorithm [12] is used to retrieve the secret. If one or more shares are modified by adversary or due to storage error then such error detection and correction capability can be used to retrieve the secret using the KGH scheme. In particular, for the (t, n)-threshold KGH scheme, more than $\lfloor (n-t)/2 \rfloor$ shares must be tampered with or in error so that the legitimate user is unable to retrieve the secret [11]. On the other hand, CLM scheme does not have any relation to Shamir's scheme and consequently Reed-Solomon decoding can not be used to detect and correct erroneous shares.

The CLM scheme like KGH scheme also uses a linear code and due to that some decoding algorithm can be used to detect and correct erroneous shares to retrieve the secret. However, it has been shown by Berlekamp *et al.* in [18] that, the decoding of an arbitrary code, e.g., codes with a random generator matrix, is NP-complete. The CLM scheme uses random generator matrix and the authors in [3] also did not present any efficient decoding algorithm for the linear code. Hence, even if more than t shares are available in the CLM scheme, it is practically impossible to detect and correct erroneous shares to recover the secret.

IV. Computational Approaches for inverse of G' in CLM Scheme

We now discuss various approaches that can be used to obtain inverse of G' matrix which is required by $Recover^{\text{IDA}}$ algorithm to obtain secret k from t pooled shares. There are mainly three ways in which this task can be performed: first, the algorithm can store precomputed inverse of all possible G' matrices. Second, inverse of G' can be calculated on the fly and third, only the first column of the inverse of G' can be calculated using a algorithm which do not invert the whole matrix. These approaches are discussed in detail in the following.

A. Precompute Inverse

The CLM (t, n)-threshold scheme can recover secret k from any t pooled shares. The decoder used in $Recover^{\text{IDA}}$ algorithm needs a $t \times t$ matrix G' which consists of the t columns of generator matrix G corresponding to the available t shares. In this approach, inverse of all possible G' are precomputed and stored so that $Recover^{\text{IDA}}$ algorithm can use

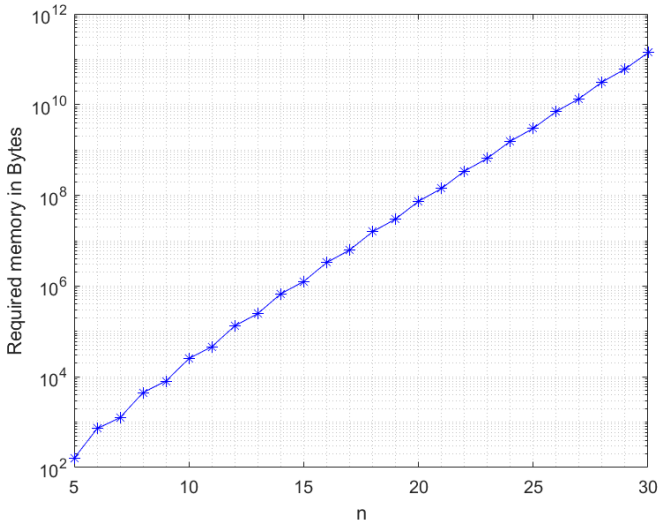

Fig. 1. Memory required in Bytes to store inverse of all possible G' for different n, here $t = \lfloor n/2 \rfloor$.

them as and when needed. For example, if $n = 6, t = 3$ then the decoder needs to store inverse of $\binom{6}{3} = 20$ 3×3 G' matrices. Since all the required inverse of matrices are available to the algorithm, this approach is the fastest among the three discussed here. However, the number of G' matrices increases exponentially as the maximum value of $\binom{n}{t}$ grows exponentially with n. The worst case occurs for $t = n/2$ when $\binom{n}{t}$ reach its maximum value. Even for moderate values of n, e.g., for $n = 30, t = 15$ the decoder requires inverse of $\binom{n}{t} = 155,117,520$ G'matrices ! Clearly, it would be very difficult to store all matrices in this case. Hence, this scheme becomes impractical even for moderate values of k and t.

On a positive side, for $n = 6, t = 3$ the algorithm needs to store 20 matrices. Since each such matrix is of size 3×3, it has 9 entries in total. Let us assume that each entry from a matrix is stored with 4 Bytes. Then, storage of an inverse would require only 36 Bytes and in total 720 Bytes of RAM is required to store 20 matrices. This memory requirement is quite low and hence this approach is preferred when n is small enough.

The memory required to store inverse of all possible G' for (t, n) threshold scheme where $t = \lfloor n/2 \rfloor$ is shown in Fig. 1. As can be observed, for $n = 26, t = 13$ the memory requirement is already 13 Gigabytes which makes this approach impractical for $n \geq 26, t = \lfloor n/2 \rfloor$.

Other possibility is to store inverse matrices in hard disk drive (HDD) and read them as and when necessary from HDD. HDDs are in general much slower than RAM and hence the read time would increase. However, since large HDDs are relative cheap, e.g., 1TB HDD cost around $ 40 US, it would be possible to store all inverse matrices for $n > 26$ using HDD. To estimate the memory required to store inverse matrices for higher n value, we use following approximation of $\binom{n}{t}$[16]

$$\binom{n}{\frac{n}{2}} \sim \sqrt{\frac{2}{m}} \frac{1}{\sqrt{n}} 2^n \tag{4}$$

where $f(n) \sim g(n)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. The

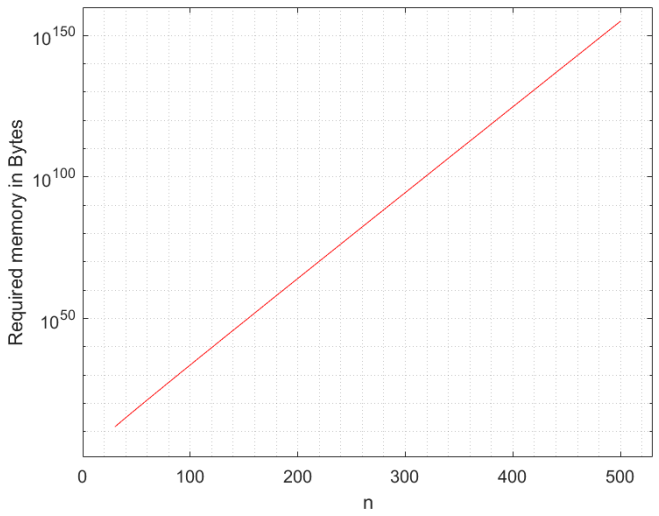

Fig. 2. Estimate of the Memory required in Bytes to store inverse of all possible G' for different n, here $t = \lfloor n/2 \rfloor$.

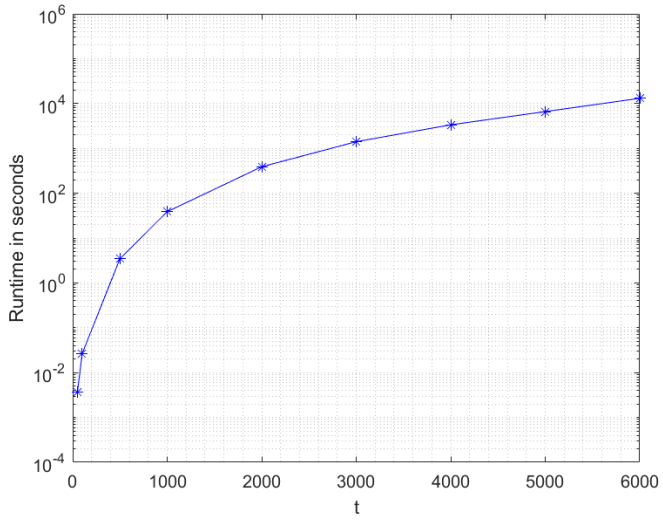

Fig. 3. Runtime in seconds to calculate inverse of $t \times t$ G' over $GF(2^4)$ for different t.

results of our calculation using (4) is shown in Fig. 2. As can be observed, the estimated memory requirement continues to grow exponentially for higher n values. For $n = 36$, our estimate shows that the required memory is around 48 Terabyte. Hence, even when HDD is used to store all inverse matrices, it is difficult to build a practical CLM scheme for $n > 36$.

B. Computing Inverse on the Fly

The other possibility to obtain inverse of G' is to calculate it on the fly in $Recover^{\text{IDA}}$ algorithm. Matrix inversion methods, e.g., Gaussian elimination, can be used for this purpose. Gaussian elimination is known to have complexity of $\mathcal{O}(n^3)$. Other possibility is to use LU decomposition to obtain lower and upper triangular matrices through which the inverse can be obtained. The complexity of LU decomposition is given by $\mathcal{O}(M(n))$ [13] where $M(n)$ is the time required to multiply two matrices of order n and $M(n) \geq n^a$ for some $a > 2$. Hence, if a faster matrix multiplication algorithm is used then the complexity of the LU decomposition can be reduced. For example, when a matrix multiplication is performed using the Coppersmith-Winograd algorithm [14] then the complexity of LU decomposition is given by $\mathcal{O}(n^{2.376})$.

To get an idea of time required to compute the inverse of G', we computed the inverse of $t \times t$ matrices over $GF(2^4)$ for different t values using NTL library [15] with a C++ program. The time required to compute the inverse for different values of t is given in Fig. 3. It can be observed that as the value of t increases, the runtime required to invert the $t \times t$ matrix increases rapidly. For $t = 6000$ the runtime required to invert matrix is already close to 4 hours and should be more than 24 hours for $t = 9000$. Though much higher values of n can be achieved through this method compared to storing of all inverse matrices, due to its high runtime requirement this method is also impractical for larger n. Further, the inversion algorithm also requires significant memory, e.g., for $n = 6000$ the NTL required about 2GB of RAM, which would also restrict the use of this method in practice.

C. Computing only the first column of the inverse

As mentioned before, the KGH scheme can be converted to Shamir's scheme by using appropriate generator matrix. Shamir's scheme has an advantage that the values of $\alpha_{i_j} - \alpha_{i_l}$ required in (2) can be precomputed and stored to accelerate the recovery process. We show in the following that, the same values can be used to calculate the first column of the inverse of the matrix G' which is derived from the t shares.

Let us assume that a secret k has been protected using KGH scheme. When only t shares, i.e., $v_{i_1}, \ldots, v_{i_t}$ are available then we get following from (3),

$$u = (v_{i_1}, \ldots, v_{i_t}) {G'}^{-1} \tag{5}$$

As can be observed from the (5), the secret $k = u_1$ can be obtained from

$$u_1 = (v_{i_1}, \ldots, v_{i_t}) {G'_{\cdot,1}}^{-1} \tag{6}$$

where ${G'_{\cdot,1}}^{-1}$ is the first column of ${G'}^{-1}$

Hence, in order to retrieve the secret $k = v_0 = u_1$ in KGH scheme, all the columns of the ${G'}^{-1}$ are not required and instead only first column of the of ${G'}^{-1}$ is enough.

As discussed in Sec. III, if G is chosen properly then KGH scheme is equivalent to the Sharmir's secret sharing scheme. The structure of G has to be selected as follows,

$$G = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & \alpha & \alpha^2 & \ldots & \alpha^{q^\lambda - 1} \\ 0 & \alpha^2 & \alpha^4 & \ldots & \alpha^{(q^\lambda - 1)^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \alpha^{k-1} & \alpha^{2k-1} & \ldots & \alpha^{(q^\lambda - 1)(k-1)} \end{bmatrix} \tag{7}$$

Then the components of v in $v = uG$ can also be evalued as

$$v_i = D(\alpha^i), \quad i = 1, 2, \ldots, n, \tag{8}$$

where

$$D(x) = u_1 + u_2 x + u_3 x^2 + \cdots + u_x^{k-1} \tag{9}$$

and $v_0 = u_1 = k$. It can be observed from (9) that KGH scheme using G from (7) is same as Shamir's scheme.

If t out of n shares are available then (2) can be used to retrieve the secret k. However, with some modifications a more efficient method can be derived which is given below [17].

$$k = \sum_{l=1}^{t} v_{i_l} \cdot \beta_l \tag{10}$$

where $v_{i_1}, \ldots, v_{i_t}$ are the t shares and

$$\beta_l = \prod_{1 \le j \le t, j \ne l} \frac{\alpha^{i_j}}{\alpha^{i_j} - \alpha^{i_l}} \tag{11}$$

Similarly, when t shares available in KGH scheme following equation can be used to retrieve the secret,

$$u = v' \cdot \tilde{G} \tag{12}$$

where $v' = (v_{i_1}, \ldots, v_{i_t})$ is the vector derived from available t shares and $\tilde{G} = {G'}^{-1}$ is derived as per Sec. II-C. However, we are interested in u_0 only and hence it is sufficient to use following equation instead,

$$u_0 = v' \cdot \tilde{G}_{.,1} \tag{13}$$

where $\tilde{G}_{.,1}$ is the first column of $\tilde{G}$. The similarities between (10) and (13) are easy to observe. The vector $\beta = (\beta_1, \ldots, \beta_t)$ is same as the column $\tilde{G}_{.,1}$. Hence, the first column of $\tilde{G}$ can be calculated using (11). Further, the computations in (11) can be accelerated by precomputing and storing $\frac{\alpha^{i_j}}{\alpha^{i_j} - \alpha^{i_l}}$ in memory. With such improvements, computation of the secret k can be accelerated substantially.

However, we remark that the above mentioned improvement can be used with KGH scheme only. On the other hand, in order to apply this method to CLM scheme it is necessary to skip the step in which XOR of secret k with random words $r_1, \ldots, r_{k-1}$ to produce k' is calculated. Instead, secret k and random words $r_1, \ldots, r_{k-1}$ should be used directly. However, as mentioned in Sec. III, with this change the CLM scheme is same as KGH scheme.

V. Applications

A. Sharing a Secret Among Board of Directors

Suppose an important document needs to be shared among n board of directors of a company in such a way that the document is accessible only when at least t members collaborate to recover it. Some (t, n)-threshold scheme can be applied to share a secret in such a scenario. According to a report, the average size of publicly traded company's board in the United Sates is 9.2 and most board size range from 3 to 31 [19]. Since the average board size is small enough, CLM scheme can be used to share the secret among board members. In such a scenario the secret can be recovered by either precomputing inverse of all possible $t \times t$ matrices G' as given in Sec. IV-A. However, when the number of board members is above 20, computing the inverse G' on the fly as discussed in Sec. IV-B may be more appropriate.

B. Boardroom Voting

Another possible use of CLM scheme is in boardroom voting in which board of directors vote in yes or no (0 or 1) on a specific proposal. Let us assume that n board members are eligible to vote and all members also act as tallying authority. Out of the total n members, at least t members must collaborate in order to count the votes. Now, each voter generates $r_1, \ldots, r_{k-1}$ random numbers over $GF(2)$ and use the CLM scheme to generate n shares from their vote which is the secret k. These n shares are then distributed among n members out of which t must come together in order to recover the vote casted by a board member. Since the number of board members is limited, both approach of obtaining inverse matrices during recover phase can be used in this case too.

C. Shareholder Voting

Each shareholder of a company has a right to vote in corporate elections. Let us assume that the shareholders of a company gather to vote on some important decision, e.g., takeover bid from another company, where they have to vote in yes or no. Here again the CLM or the KGH secret sharing scheme can be used to conduct elections. This application is very similar to the previous one however, the number of shareholders of a company are likely to be much higher than board of directors. Hence, the number of shares, i.e., n, generated by each voter along with the minimum number of shares, i.e., t required to recover a vote could be much higher. If we assume n in order of several thousand then the CLM scheme can be used along with recover algorithm which calculates inverse of G' on the fly.

In the above two examples if a member of tallying authority modifies a share then as explained in Sec. III, it is practically impossible to detect and correct such modified shares during recovery phase using the CLM scheme. On the other hand, if the KGH scheme is used along with an appropriate generator matrix then Reed-Solomon decoding can be used to detect and correct modified shares. Hence, the KGH scheme is more appropriate in both the schemes when tallying authority can not be trusted.

VI. Conclusion

In this paper, we analyzed the (t, n)-threshold secret sharing scheme proposed by Chen *et al.* in CANS 2016. First, we showed that this scheme is very similar to the one proposed earlier by Karnin *et al.* in 1983. Then, we made three more observations: 1) the authors in [3] did not consider the memory and XOR operations needed for obtaining an inverse matrix required during recovery of the secret, 2) XOR operations needed to compute matrix and vector multiplication for the secret recovery are also not considered in their analysis, 3) since the CLM scheme lack efficient decoding algorithm, it can not detect or correct erroneous shares whereas the KGH scheme can be designed for the same. Further, we proposed and discussed three methods to obtain inverse matrix. We conclude that the CLM scheme is practical for $n \le 30$ when all possible inverse matrices are stored in memory whereas up to $n = 9000$ can be obtained if the inverse matrix is calculated on the fly. The third method of obtaining only the first column of the inverse matrix through Lagrange's interpolation can be

used only with the KGH scheme. The CLM scheme becomes impractical if n is of the order of tens of thousands.

REFERENCES

[1] A. Shamir, "How to share a secret," *Communications of ACM*, 22(11), pp. 612–613, 1979.

[2] G. Blakely, "Safeguarding cryptographic keys," In *Proc. of the National Computer Conference*, vol. 48, pp. 313–317, 1979.

[3] L. Chen, T. M. Laing, and K. M. Martin, "Efficient, XOR-Based, Ideal (t, n)-threshold Schemes," in *Proc. International Conference on Cryptology and Network Security, CANS 2016*, pp. 467–483, Milan, Italy, November 14-16, 2016.

[4] E. Karnin, J. Greene, and M. Hellman, "On secret sharing systems," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 35–41, January 1983.

[5] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A new (k, n)-threshold secret sharing scheme and its extension," In : Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 455470. Springer, Heidelberg (2008).

[6] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A fast $(3, n)$-threshold secret sharing scheme using exclusive-or operations," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 91(1), pp. 127–138, 2008.

[7] Lv, C., Jia, X., Tian, L., Jing, J., Sun, M., "Efficient ideal threshold secret sharing schemes based on exclusive-or operations," In *Proc. 4th International Conference on Network and System Security* (NSS), pp. 136–143, 2010.

[8] Lv, C., Jia, X., Lin, J., Jing, J., Tian, L., Sun, M., "Efficient secret sharing schemes," In: Park, J.J., Lopez, J., Yeo, S.-S., Shon, T., Taniar, D. (eds.) Secure and Trust Computing, Data Management, and Application. Communications in Computer and Information Science, vol. 186, pp. 114121. Springer, Heidelberg (2011).

[9] Y. Wang, Y. Desmedt, "Efficient secret sharing schemes achieving optimal information rate," In *Proc. IEEE Information Theory Workshop (ITW) 2014*, pp. 516–520, Tasmania, Australia, November 2014.

[10] Chen, L., Camble, P.T., Watkins, M.R., Henry, I.J., "Utilizing error correction (ECC) for secure secret sharing," Hewlett Packard Enterprise Development LP, World Intellectual Property Organisation. Patent Number WO2016048297 (2016). https://www.google.com/patents/WO2016048297A1?cl=en.

[11] R. J. McEliece, and D. V. Sarwate, "On sharing secrets and Reed-Solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583-584, September 1981.

[12] E. R. Berlekamp, (1984) [1968], *Algebraic Coding Theory* (Revised ed.), Laguna Hills, CA: Aegean Park Press, ISBN 0-89412-063-8. Previous publisher McGraw-Hill, New York, NY.

[13] J. R. Bunch and J. E. Hopcroft, "Triangular Factorization and Inversion by Fast Matrix Multiplication," Mathematics of Computation, vol. 28, no. 125, pp. 231–236, Jan., 1974.

[14] D. Coppersmith, S. Winograd, "Matrix multiplication via arithmetic progressions," vol. 9, no. 3, pp. 251–280, March 1990.

[15] NTL: A Library for doing Number Theory, http://www.shoup.net/ntl/

[16] T. Worsch, "Lower and Upper Bounds for (Sums of) Binomial Coefficients," available online, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.9677

[17] A. Beimelm, "Secret-Sharing Schemes: A Survey," in Proc. International Conference on Coding and Cryptology (IWCC 2011), pp. 11–46, Qingdao, China, May-June, 2011

[18] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," IEEE Transactions on Information Theory, vol. 24, no. 3, May 1978.

[19] https://en.wikipedia.org/wiki/Board_of_directors#Size_2