European Wireless 2013, 16 – 18 April, 2013, Guildford, UK ISBN 978-3-8007-3498-6 © VDE VERLAG GMBH • Berlin • Offenbach, Germany

A Probabilistic Algorithm for Secret Matrix Share Size Reduction

Eckhard Pfluegel*, Emmanouil Panaousis† and Christos Politis*

*Wireless, Multimedia & Networking (WMN) Research Group
Kingston University London
{e.pfluegel, c.politis}@kingston.ac.uk

†Queen Mary, University of London
panaousis@eecs.qmul.ac.uk

***Abstract*—Secret sharing is an important tool in cryptography and has many applications for wireless networks. This paper is motivated by the need for space-efficient secret sharing schemes. We first propose a simple probabilistic algorithm which can be used, prior to secret sharing, in order to split a given secret into public and private data. The public data can be made openly available, and any specific secret sharing method can be used in order to share the private data. We then show that, combined with a previously published space-efficient single secret sharing method, this yields a novel probabilistic matrix-based online multi-secret sharing method with small expected share size. In particular, compared with other matrix-based approaches, our scheme is of similar expected computational cost but smaller share size. Finally, we report on an implementation of our method and evaluate its performance. Our algorithm could be useful to design efficient secret sharing applications for wireless networks, in particular mobile ad-hoc networks, in areas such as secure routing, data transmission or key management.**

***Index Terms*—Space-Efficient Multi-Secret Sharing, Probabilistic Algorithms**

I. Introduction

Due to vulnerabilities inherent of the wireless medium and the mobile nature of modern smart devices, the cryptographic technique of secret sharing can help distributing user and protocol data, information related to key management, and trust in general [1]. For example, this becomes particularly crucial in decentralised and autonomous wireless networks due to the absence of a trusted authority which could act as the main provider of cryptographic material for network participants. Sharing sensitive information across several nodes in such peer-to-peer or mobile ad-hoc networks (MANETs) can help improving confidentiality, integrity and availability.

Secret sharing is based on the idea that data (the *secret*) can be protected by dividing it into several parts (the *shares*) and distributing these shares to individual parties (the *shareholders*). The original data can then be deleted. Knowledge of a single share will not help with reconstructing the secret. A (k, n) *perfect secret sharing scheme* implements the following properties:

- It divides the secret into n shares.
- Any coalition of $k \leq n$ shareholders is able to reconstruct the secret.
- Any coalition of less than k shareholders cannot reveal any partial information on the secret.

Over the years, since it was introduced independently by Shamir [2] and Blakley [3], a great number of different secret sharing techniques and variants have been introduced in the literature and additional properties and features have been developed, see e.g. [4] and the references therein.

In the context of secret sharing implementations for wireless security applications [5], a focus on lightweight algorithms has been made in recent research. As it is important to minimise network traffic overhead and energy costs of resource-constrained mobile devices, techniques have been developed which are particularly efficient when dealing with several (or equivalently, large) secrets. Amongst these, due to their good performance, particular attention should be given to the class of matrix-based approaches as initiated by Bai ([6], [7]) and subsequently extended in [8]. A matrix-based approach was also undertaken in the sophisticated scheme by [9], however not focussing on space-efficiency.

In this paper, inspired by the matrix-based approach, we present two contributions:

- We propose a simple probabilistic algorithm which can be used, prior to secret sharing, in order to split a given secret s into public data P and private data Q where $|P| = |s|$ and the expected size of the private data is $|Q| = \Theta(\sqrt{|s|})$. The public data P can be made openly available, and any specific secret sharing method can then be used in order to share the private data Q. We refer to this method as a *Matrix Share Size Reduction* (MSSR) technique. We have implemented our MSSR technique and report on some tests we have carried out.
- We show that combined with a space-efficient single secret sharing method this yields a probabilistic online multi-secret sharing method with expected small share size. For example, when using the space efficient (k, n) secret sharing scheme from [10], we obtain for a share s_i the expected share size

$$|s_i| = \frac{1}{k-1} \cdot \sqrt{\Theta(|s|)}$$

where $|s|$ denotes the size of the original secret. Compared to the methods presented by ([6], [8]), this is an improvement by a factor $k-1$ which, depending on the value of k, can be substantial.

This paper is organised as follows. In Section II we introduce some notations and terminology. In Section III, our algorithm is given while Section IV analyses its space-efficiency, cost as well as security. Section V explains the use of the algorithm as a MSSR for a space-efficient online multi-secret sharing scheme. A description of our implementation and evaluation of the algorithm is given in Section VI, followed by the conclusion of this paper.

II. Notations and Terminology

Let us consider a secret s, to be distributed into n different shares s_i ($1 \leq i \leq n$), so the knowledge of at least k ($1 < k \leq n$) shares is necessary and sufficient to reconstruct the initial secret s. *Online multi-secret sharing* is a specialised secret sharing technique that aims at efficient sharing of secrets with the help of additional public data.

A *cyclic* square matrix $A \in \mathbb{Z}_p^{t\times t}$ is a matrix whose minimal polynomial and characteristic polynomial coincide. Any cyclic matrix is similar to its so-called *companion form* [11]. This form is a matrix C with all entries zero except for the elements in the upper off-diagonal which are ones, and arbitrary elements in the bottom row. Hence, the matrix can be written as

$$C = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \\ c_0 & c_1 & \dots & c_{t-2} & c_{t-1} \end{pmatrix}. \quad (1)$$

The coefficients of the characteristic polynomial of the matrix

$$f(\lambda) = \det(A - \lambda I) = \lambda^t - c_{t-1}\lambda^{t-1} - \dots - c_1\lambda - c_0 \quad (2)$$

can be read from the elements in the bottom row of the matrix (1).

In this paper, we design a probabilistic algorithm that always terminates, but does not necessarily compute the desired result. This is commonly referred to as a *Monte Carlo probabilistic algorithm*.

All scalar and matrix arithmetic in this paper is done in the field $\mathbb{Z}_p$. We will state this use of modular arithmetic explicitly in our presented pseudo-code, but will otherwise always assume that arithmetic is done modulo p.

For an integer d, we denote by $|d| = \log_b s$ its size. For example, we have for $b = 2$ that $|d| = \log_2 d$, the number of bits required to store d.

III. The Algorithm

In this section, we will explain our matrix share size reduction algorithm, and illustrate it using an example.

A. Conversion of Secret Scalar to Secret Matrix

Many secret sharing techniques commonly handle secrets as integers. From a mathematical point of view however, integers do not possess much structure. The idea that appeared in [8] and later also in [9] is to convert an integer secret to a more complicated mathematical object (an integer matrix) and to exploit the resulting new properties. We find this beneficial as well, and this conversion is the first step for our algorithm.

The following algorithm takes as input the given secret scalar value s, and a prime number p satisfying

$$p \leq \sqrt{s}. \quad (3)$$

It then converts s into a matrix $S \in \mathbb{Z}_p^{t\times t}$ where

$$t = \left\lceil \sqrt{1 + \lfloor \log_p(s) \rfloor} \right\rceil. \quad (4)$$

This is achieved by writing s as a base p number and populating the matrix S with its digits. We aim at choosing p so that we obtain a square radicand in (4), but if this is not possible, we have to pad with additional zeros. The condition (3) ensures $t \geq 2$.

The algorithm can be described as follows:

Algorithm 1 ConvertSecret(s, p)

1: Choose a prime number $p \leq \sqrt{s}$
2: Compute the digits of s as a number to the base p
3: Arrange these digits in a $t \times t$ matrix S (by filling in zero elements if necessary)
4: Return the matrix S

B. Matrix Share Size Reduction

The idea is now to achieve data compression by computing a canonical matrix normal form of the secret matrix S and to extract its characteristic data. In our algorithm, we wish to compute a companion matrix, similar to S, and retain the characteristic polynomial of the matrix. This is more efficient than computing the Jordan Normal Form as suggested in [9], since it does not require field extensions of $\mathbb{Z}_p$. However, we can only give a probabilistic algorithm, as not every matrix is cyclic. The algorithm takes as input the secret matrix $S \in \mathbb{Z}_p^{t\times t}$ and prime number p, and returns either FALSE or the characteristic polynomial of S together with a similarity transformation W bringing S into companion form.

Remark 3.1: This algorithm will successfully compute a companion matrix, provided both of the following assumptions hold:

(i) The matrix S is a cyclic matrix.
(ii) The vector v chosen at random in Step 1 of Algorithm 2 is a cyclic vector.

The idea of our share size reduction method is now to repeatedly choose random prime numbers and to convert the secret to a matrix, followed by an attempt to compute a companion form. In the next section, we will show that a maximal share size reduction is achieved for small values of p. Hence, we start with $p = 2$ and successively find the next

Algorithm 2 ComputeCompanionMatrix(S, p)

v := random row vector $\in \mathbb{Z}_p^{1\times t}$
$w_1 := v$
for $i = 2$ to t **do**
  $w_i := w_{i-1} \cdot S \bmod p$
end for
W := matrix whose rows are made from the w_i
if $\det W \equiv 0 \bmod p$ **then**
  return *FALSE*
end if
$C := WSW^{-1} \bmod p$
Return W and the bottom row of C

prime greater than p. The function ReduceShareSize expects the secret s as input and returns as public data $P \in \mathbb{Z}_p^{t\times t}$ the computed similarity transformation. The private data Q consists of the coefficients of the characteristic polynomial f and the prime number p used for the size reduction.

Algorithm 3 ReduceShareSize(s)

success := FALSE
$p := 2$
while success = FALSE and $p \le \sqrt{s}$ **do**
  S := ConvertSecret(s, p)
  success := ComputeCompanionMatrix(S, p)
  p := next_prime(p)
end while
if success $\neq$ FALSE **then**
  Retrieve the similarity transformation W and the characteristic polynomial f
  Return W as public data and the coefficients of f, together with p, as private data
else
  Return FAIL
end if

C. Example

In order to illustrate our method, we give an example using the secret $s = 123456789123456789$. The algorithm ComputeCompanionMatrix did not succeed with the choice $p = 2$, but converts s to the matrix below using $p = 3$:

$$S = \begin{pmatrix} 0 & 0 & 1 & 2 & 0 & 2 \\ 0 & 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 & 2 & 2 \\ 2 & 0 & 2 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 0 & 1 & 1 & 2 \end{pmatrix}. \tag{5}$$

As public data, we obtain the transformation

$$W = \begin{pmatrix} 1 & 2 & 1 & 0 & 2 & 0 \\ 0 & 1 & 2 & 1 & 2 & 2 \\ 0 & 2 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 2 & 2 & 0 & 1 & 0 & 0 \\ 2 & 2 & 2 & 2 & 0 & 0 \end{pmatrix}. \tag{6}$$

The matrix $C = WSW^{-1}$ is in companion form:

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 & 1 & 0 \end{pmatrix} \tag{7}$$

It is now sufficient to share the coefficients $[0, 0, 2, 2, 1, 0]$ of the characteristic polynomial as well as p.

IV. Analysis of our MSSR Technique

We analyse the likelihood of success, space and cost requirements of our MSSR technique as well as discuss its security.

A. Space and Cost — Worst Case

When inspecting the space-efficiency of our MSSR technique, we need to compute the sizes of the public data P and private data Q. We see immediately $|P| = |W| = |S| = |s|$. Also, $|Q| = |f| + |p|$ where

$$|f| = t \cdot |p| = \frac{|S|}{t} = \frac{|s|}{t}$$

where t as in (4). Hence,

$$|f| = \frac{|s|}{\left\lceil \sqrt{1 + \lfloor \log_p s \rfloor} \right\rceil} = \frac{|s|}{\Theta(\sqrt{|s|})} = \Theta(\sqrt{|s|}).$$

A more refined analysis yields

$$\frac{1}{\left\lceil \sqrt{1 + \lfloor \log_p s \rfloor} \right\rceil} \le \frac{1}{\sqrt{\log_p s}}$$

hence

$$|f| \le \sqrt{\frac{\log_b p}{\log_b s}} \cdot |s| = \sqrt{|p|} \cdot \sqrt{|s|}.$$

From this follows the size estimate for the private data

$$|Q| = \Theta(\sqrt{|s|}) + |p| = \Theta(\sqrt{|s|}) + \log_b(\sqrt{s}) = O(|s|)$$

in the worst case. In this case, we do not achieve a reduction in share size.

Concerning the worst-case cost analysis of the algorithm, we observe that Algorithm 2 requires $t - 1$ multiplications of a matrix vector by matrix, followed by a modular determinant

computation, matrix inversion and two matrix multiplications. All vectors and matrices are of dimension t. Algorithm 3 calls Algorithm 1 and 2 at the most l times where $l = \pi(x)$ is the number of primes less or equal than $x = \sqrt{s}$. Using the fact that for large values of x, the prime number theorem states $\pi(x) \sim \frac{x}{\ln x}$, we obtain

$$l = O\left(\frac{\sqrt{s}}{\log \sqrt{s}}\right) = O\left(\frac{b^{|s|/2}}{|s|}\right) = O(b^{|s|}).$$

Overall, this is an exponential worst-case run-time as a function of the input size.

B. Expected Space and Cost

In order to assess the expected (average) space and cost requirements of our algorithm, we will assert that its probability of success, using a small prime number p, is relatively high. The underlying reason is that there are enough cyclic matrices and vectors available in $\mathbb{Z}_p^{t\times t}$.

Lemma 4.1: The probability that Algorithm 2 successfully computes a companion matrix for a secret matrix S and a given prime p is

$$\text{Prob[companion}_p\text{]} = (1-p^{-1})(1-p^{-2})\cdots(1-p^{-t})$$

where t is defined as in (4).

Proof The proof of this lemma is straightforward by using the result of [11, Proposition 2.8] which says that the probability that the two assumptions (i) and (ii) in Remark 3.1 hold simultaneously is exactly $(1-p^{-1})(1-p^{-2})\cdots(1-p^{-t})$. □

An estimate can be given for Prob[companion$_p$] that is independent of t. We recall that $t \geq 2$.

Lemma 4.2: We have the following probability estimate:

$$\text{Prob[companion}_p\text{]} > 1 - p^{-1} - p^{-2}.$$

Proof This follows from Lemma 4.1 and, since $t \geq 2$, the estimate

$$(1-p^{-1})(1-p^{-2})\cdots(1-p^{-t}) > 1 - p^{-1} - p^{-2}$$

which is given in [11, Lemma 3.5]. □

We see that if $p > 2$, the estimate gives Prob[companion$_p$]> 0.5. Repeated iteration of Algorithm 1 and Algorithm 2 allows then increasing the probability of success to any desired value.

Lemma 4.3: The probability that Algorithm 3 terminates successfully after at the most $l+1$ iterations ($l \geq 2$) is

$$\text{Prob[success}_l\text{]} > 1 - \frac{3\cdot 4^{l-2}}{9^{l-1}}.$$

Proof We compute the desired probability using the formula

$$\text{Prob[success}_l\text{]} = 1 - \prod_{i=1}^{l}(1 - \text{Prob[companion}_{p_i}\text{]}).$$

We can now use Lemma 4.2 in order to estimate this probability as

$$\begin{aligned}\text{Prob[success}_l\text{]} &> 1 - (2^{-1}+2^{-2})(3^{-1}+3^{-2})^{l-1} \\ &= 1 - \frac{3\cdot 4^{l-2}}{9^{l-1}}.\end{aligned}$$

□

For example, we have for $l = 9$ that Prob[success$_9$] ≈ 0.999, so on average only one in 1000 attempts would need more than 10 iterations.

We summarize our space and cost analysis with the following

Theorem 4.1: Algorithm 3 is a probabilistic Monte Carlo algorithm with an exponential worst-case run time, as a function of the size of the input s. The expected reduced share size is $\Theta(\sqrt{|s|})$.

Proof The statement about the worst-case analysis follows from the considerations of the previous section. Lemma 4.3 shows that the expected number of iterations is small. As a consequence, the size of p will be negligible, and the expected size of the private data is then $|Q| = |f| = \Theta(\sqrt{|s|})$. □

C. Security

Assume an adversary wants to exploit the availability of the public data P in order to derive the secret matrix S. He knows that P transforms S into a companion matrix C, but he can only guess its value. The size of the search space $\mathcal{S}$ is

$$|\mathcal{S}| = |Q| = \Theta(\sqrt{|s|}).$$

Thus, reducing the share size reduces the security as we have a smaller search space. Also, the elements of the search space might not be equally distributed. This is the price to pay for the increased space efficiency. However, it is clear that knowing P does not give any advantage in deducing the secret matrix S.

V. Use for Online Multiple-Secret Sharing with Small Shares

Combined with a particular single secret sharing scheme, we can use our MSSR technique in order to obtain a probabilistic online multi-secret sharing scheme.

After calling Algorithm 3, shares can be created by sharing Q using any (k, n) secret sharing scheme. We share the coefficients of f and the value of p, and store the similarity transformation W as public information. We detail this in Algorithm 4.

The shares s_i have to be distributed securely in order to guarantee data confidentiality and integrity. The public data P needs to be authenticated.

The secret can be reconstructed from a subset $\mathcal{A} = \{s_{i_1}, \ldots, s_{i_k}\}$ of k shares and the public information P as described in Algorithm 5.

Algorithm 4 CreateShares(f, k, n)

1: success := ReduceShareSize(s)
2: **if** success $\neq$ FALSE **then**
3: Let SSS be a (k, n) secret sharing scheme
4: Create shares $s_1, \ldots, s_n$ by sharing f and s using SSS
5: Return $s_1, \ldots, s_n$
6: **end if**

Algorithm 5 ReconstructSecret($\mathcal{A}$, P)

1: Reconstruct the characteristic polynomial f and the value of p by acquiring at least k shares.
2: Build C from the t coefficients of f.
3: Use the public data P in order to compute the secret matrix $S = P^{-1}CP \bmod p$.
4: Reconstruct s from S and p.

Now suppose we use in Algorithm 4 the method from [10] as secret sharing scheme. This method creates shares the sizes of which are the size of the original secret divided by $k-1$. Overall, with the expected size of Q being $|Q| = \Theta(\sqrt{|s|})$, we hence obtain the expected share size

$$|s_i| = \frac{1}{k-1} \cdot \sqrt{\Theta(|s|)}.$$

Bai's method was originally formulated taking as input t^2 secrets arranged in a matrix S, and it creates shares that are vectors of t elements. Hence, it only achieves a share size of $\sqrt{\Theta(|s|)}$.

In terms of expected computational cost, our scheme seems similar to Bai's deterministic cost. The main theoretical disadvantage of our method is its probabilistic nature, although in practice we believe that this is not a problem. It is important to realise that our main contribution is the matrix share size reduction technique which in itself does not really compare with Bai – the latter integrating both size reduction as well as share creation into one method.

VI. Implementation and Evaluation

We describe an implementation of our algorithm that we have carried out using the computer algebra system Maple. We evaluate our implementation by running the algorithm for a large sample of secrets.

A. Description

Our algorithm needs efficient handling of linear algebra operations modulo a prime number p. After experimenting with several programming languages, we decided to adopt Maple for our implementation. The Maple *Modular* package provides efficient modular arithmetic for tasks such as computing determinants, matrix products and inverses. This enabled us to write a concise and efficient program – our initial implementation of Algorithm 1, 2 and 3 contains only 51 lines of code and can be downloaded at [12].

However, Maple being a symbolic computational tool, it tends to be slower than numerical packages such as Matlab or mainstream compiled programming languages. Hence, it may be possible to further improve the execution times that we will be reporting in the next section. A more sophisticated implementation, for example in C/C++ or Java, using suitable maths libraries, would undoubtedly be desirable if speed was an issue.

For our study, we will focus on evaluating the behaviour of our randomised algorithm focussing on its probability of success, and the achieved share size reduction.

B. Evaluation

For secrets with various sizes $10 \leq |s| \leq 1000$ where $|s| = \log s$ is the number of decimal digits of s, we have created a sample of 1000 secrets each. For each sample, we have run the algorithm, recording l_{min}, l_{max} and l_{avg} as the minimal, maximal and average number of iterations needed to achieve a share size reduction. We have also given the average execution time per sample t_{exec}, measured in seconds. From the information on l_{avg}, one can derive the average value of the prime number p used during the algorithm, as well as the achieved share size reduction.

TABLE I
MSSR Evaluation for Sample of Shares

$\lvert s\rvert = \log s$	l_{min}	l_{max}	l_{avg}	t_{exec}
10	1	5	2.29	0.905
20	1	4	2.52	1.232
30	1	5	2.3	1.326
40	1	6	2.46	1.716
50	1	5	2.28	1.997
100	1	5	2.7	3.79
200	1	9	2.32	8.144
300	1	5	2.42	13.65
400	1	5	2.32	19.765
500	1	6	2.32	26.833
1000	1	5	2.06	70.06

This data confirms that the average number of iterations l_{avg} seems independent of t, hinted at by our estimate in Lemma 4.1. Generally, l_{avg} is small (≤ 5 in the majority of size samples), the algorithm successfully terminated after only one iteration at least once in each sample, and never needed more than 9 iterations. Hence, the biggest prime number that was used was $p = 19$, very much in line with our estimate of Lemma 4.3.

VII. Conclusion

In this paper, we have given a probabilistic algorithm to achieve matrix share size reduction. This algorithm can be used, combined with single secret sharing methods, in order to implement space-efficient online multi-secret sharing schemes. We have derived the expected and worst-case space-efficiency and computational cost of our algorithm, and have showed how to obtain a scheme that is more space-efficient than previously

published matrix-based methods. We have evaluated our algorithm, using an implementation, and show that the average space-efficiency and cost confirm our theoretical analysis.

Several pieces of future work may arise from this paper. We would like to more systematically implement and evaluate the entire secret sharing mechanism including share creation. This would need an implementation of the method in [10]. An implementation of Bai's method and comparison with our method would also be valuable. Furthermore, we could apply our MSSR method recursively, by applying it to the private data Q. This would further reduce the size of the private data, but augment the amount of public data one would have to provide. Also, extending our algorithm for the use in dynamic, proactive or verifiable secret sharing could be envisaged. This could be useful for the design of efficient secret sharing applications for wireless networks. Especially, architectures that consider *distributed hash tables* for mobile ad-hoc networks, as the one proposed in [13] and [14], could take advantage of our algorithm to improve network security. Last but not least, we would like to design a deterministic algorithm that systematically performs as well as our probabilistic algorithm does on average.

REFERENCES

[1] G. Polymerou, E.A. Panaousis, E. Pfluegel and C. Politis, "A Novel Lightweight Multi-secret Sharing Technique for Mobile Ad-hoc Networks," *Proc. of the 29th Wireless World Research Forum (WWRF), Berlin, Germany*, October, 2012.

[2] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[3] G. R. Blakley, "Safeguarding cryptographic keys," *Managing Requirements Knowledge, International Workshop on*, vol. 0, p. 313, 1979.

[4] A. Beimel, "Secret-sharing schemes: A survey," in *Coding and Cryptology*, ser. Lecture Notes in Computer Science, Y. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, Eds. Springer Berlin Heidelberg, 2011, vol. 6639, pp. 11–46.

[5] H. Deng, A. Mukherjee, and D. Agrawal, "Threshold and identity-based key management and authentication for wireless ad hoc networks," *in Proc. International Conference on Information Technology: Coding and Computing (ITCC)*, vol. 1, pp. 107–111, Apr. 2004.

[6] L. Bai, "A strong ramp secret sharing scheme using matrix projection," *in Proc. International Symposium on World of Wireless, Mobile and Multimedia Networks (WOWMOM), IEEE Computer Society*, pp. 652–656, 2006.

[7] L. Bai and X. Zou, "A proactive secret sharing scheme in matrix projection method," *International Journal of Security and Networks, Inderscience*, vol. 4, no. 4, pp. 201–209, 2009.

[8] K. Wang, X. Zou, and Y. Sui, "A multiple secret sharing scheme based on matrix projection," *in Proc. Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 400–405, Jul. 2009.

[9] D. Zhao, H. Peng, C. Wang, and Y. Yang, "A secret sharing scheme with a short share realizing the threshold and the adversary structure," *Computers & Mathematics with Applications*, vol. 64, no. 4, pp. 611 – 615, 2012.

[10] A. Parakh and S. Kak, "Space efficient secret sharing for implicit data security," *Information Sciences*, vol. 181, no. 2, pp. 335 – 341, 2011.

[11] P. M. Neumann and C. E. Praeger, "Cyclic matrices over finite fields," *Journal of the London Mathematical Society*, vol. 52, no. 2, pp. 263–284, 1995.

[12] E. Pfluegel, "Matrix share size reduction." [Online]. Available: http://staffnet.kingston.ac.uk/~ku32104/MSSR

[13] G.P. Millar, E.A. Panaousis and C. Politis, "ROBUST: Reliable overlay based utilisation of services and topology for emergency MANETs," *in Proc. Future Network and Mobile Summit, IEEE, Florence, Italy*, Jun.16-18, 2010.

[14] G.P. Millar, E.A. Panaousis and C. Politis, "Distributed Hash Tables for Peer-to-Peer Mobile Ad-hoc Networks with Security Extensions," *Journal of Networks*, vol. 7, no. 2, pp. 288-299, issn: 1796-2056, February 2012.