

Recursive Secret Sharing for Distributed Storage and Information Hiding

Abhishek Parakh and Subhash Kak

Computer Science Department, Oklahoma State University, Stillwater, OK 74078 USA

Emails: {parakh, subhashk}@cs.okstate.edu

Abstract—This paper presents a recursive computational multi-secret sharing technique that hides $k - 2$ secrets of size b each into n shares of a single secret S of size b , such that any k of the n shares suffice to recreate the secret S as well as all the hidden secrets. This may act as a steganographic channel to transmit hidden information or used for authentication and verification of shares and the secret itself. Further, such a recursive technique may be used as a computational secret sharing technique that has potential applications in secure and reliable storage of information on the Web, in sensor networks and information dispersal schemes. The presented technique, unlike previous computational techniques, does not require the use of any encryption key or storage of public information.

I. INTRODUCTION

An information theoretically secure k -out-of- n secret sharing technique used to share a secret of size b requires a total storage space of size $b \cdot n$. Since, $k - 1$ shares do not reveal any information about the secret, such techniques use $k - 1$ random elements of size b in order to create the shares. In this paper, we propose that these random elements be replaced with certain hidden information that may serve as a steganographic channel. Note that if a secret sharing scheme uses $k - 1$ random elements then $k - 1$ is the upper limit on the number of secrets that can be hidden. We hide $k - 2$ secrets which is near optimal.

For example, if user A transmits a secret message to user B over a public channel, he may divide the message into several pieces (some of them redundant) and send the pieces on parallel channels over the network. Further, the pieces may be created using a secret sharing technique such that an eavesdropper may need to compromise (or listen to) to at least k channels in order to recreate the message from the pieces. It is conventional that B upon receiving the pieces may reconstruct the message and authenticate it using the signed hash of the message that A sends to B . Transmission of this signature is an additional burden on the network. In the proposed scheme, A may hide the signature within the pieces of the message that is transmitted.

Information dispersal schemes are growing in importance for distributed storage networks and implementations have appeared at CMU and IBM [1], [2], [3]. These systems primarily use computational secret [3], [4], [5] sharing schemes. In general, in a computational secret sharing scheme an encryption key is used to encrypt the secret/message that is to be securely stored/transmitted. The encrypted message is then divided into several (possibly redundant) pieces. The key is divided into shares using a conventional secret sharing technique and the shares of this key are stored along with the pieces of the encrypted message [6], [7], [8]. The shares of the key become

an overhead, especially, if the information dispersal scheme is to be used in sensor networks. In order to recover the secret in presence of wrong shares, robustness was added using hash-functions in [9] and general implementations of computational secret sharing appeared in [5], [4].

In a multiparty scenario, such as in secret sharing, the hidden information may be used as a means of authentication of the share (and of reconstructed secret), thus provide cheating detection. For example, the dealer may hide a “specially” chosen message in the shares of the secret and distribute the hash of this message to all the players along with the shares. The players may later reconstruct the secret and the hidden message, find the hash of the hidden message and verify it against the hash they have.

In this paper we present a multi-secret sharing scheme that uses Shamir’s secret sharing scheme as its building block and encodes $k - 2$ additional secrets within the shares of the message originally intended to be shared. The scheme may be used as a computational secret sharing scheme, effectively resulting in smaller shares, by dividing a secret into smaller pieces and then simulating a multi-secret sharing scheme. Since the proposed algorithm generates shares on the order of size of secrets encoded, smaller pieces will give rise to smaller shares. The proposed scheme does not require any encryption key.

An efficient method for sharing multiple secrets with security based on assumption of hardness of discrete logarithm problem is presented in [10]. Whereas [11] proposes a scheme based on systematic block codes and [12] propose schemes based on Shamir’s secret sharing scheme but require a large amount of side information to be stored as public knowledge and further [11], [12], [13] attempt to maintain ideal security. Other schemes [14], [15] focus at improving efficiency of computations involved in share creation and secret reconstruction rather than space efficiency and transmission efficiency.

In an earlier paper [16], a 2-out-of-2 ($k = 2$ and $n = 2$) recursive scheme for secret sharing was proposed. In this method, if k secrets are chosen such that they double in size, then all of the smaller secrets can be recursively stored in the shares of larger secrets, so that two shares of size 2^m can encode $2^{m+1} - 1$ bits of information. For example, if we are to share 3 secrets $s_1 = 1$, $s_2 = 01$, and $s_3 = 1011$, then the two shares for s_1 would be $D_{s_11} = 0$ and $D_{s_12} = 1$; where exclusive-OR operation is used for secret reconstruction. The shares of s_1 can be used to create two shares of s_2 as follows: $D_{s_21} = D_{s_11}0 = 00$ and $D_{s_22} = 0D_{s_12} = 01$. Here $D_{s_11}0$ denotes concatenation of share 1 of secret s_1 with 0; and $0D_{s_12}$ denotes concatenation of 0 with share 2 of secret

s_1 , and so on. Similarly, we can recursively use the shares of s_2 to create the shares of s_3 : $D_{s_3 1} = D_{s_2 1} 10 = 0010$ and $D_{s_3 2} = 10 D_{s_2 2} = 1001$. As a result, the final two shares for all the three secrets are 0010 and 1001. Consequently, using 8 bits of shares we have encoded 7 bits of secrets. This is in comparison with conventional methods that would require a total 14 bits of shares. The of recursive secret sharing has been extended to 2-out-of- n secret sharing in [17].

The above efficiency increase is obtained as a tradeoff against security of the scheme. For example, a conventional (non-recursive) implementation would require 7 bits for each share but since the recursive implementation only requires 4 bits for each share, a player only needs to determine 4 bits to break the scheme. However, in practise often secrets are thousands of bits long. For example, a secret of 1048 bits length would be encoded in approximately 1024 bits per share, and would still require 2^{1024} combinations to break. This may be sufficient for many cases.

II. SPACE EFFICIENT SECRET SHARING

We propose a method to hide $k-2$ secrets of size b , within the shares of a secret S of size b , using a (k, n) modified Shamir's secret sharing scheme. The secret is divided into n shares using modified Shamir's secret sharing scheme such that any k of them can be brought together for reconstruction.

A modified Shamir's scheme is presented in Algorithm 1.

Algorithm 1 (Modified Shamir's secret sharing scheme)

- 1) Choose a prime $p, p > \max(S, n)$, where S is the secret.
- 2) Choose $k-1$ random numbers y_1, y_2, \dots, y_{k-1} , uniformly and independently, from the field \mathbb{Z}_p .
- 3) Map these random numbers y_i s as y coordinates of points: (i, y_i) , for all $1 \leq y_i \leq (k-1)$.
- 4) Map the secret S as point $(0, S)$.
- 5) Using k points (i, y_i) , for all $1 \leq y_i \leq (k-1)$ and $(0, S)$ interpolate a polynomial $p(x)$ of degree $k-1$ modulo prime p .
- 6) Sample $p(x)$ at n points $D_i = p(i), k \leq i \leq k+n-1$ such that the shares are given by (i, D_i) .

The reconstruction procedure for the secret follows the conventional method [18].

Now consider $k-2$ secrets $s_1 s_2 \dots s_{k-2}$, $s_i \in \mathbb{Z}_p$ for all $1 \leq i \leq (k-2)$. Therefore our task is to recursively hide s_i 's within the shares of secret S . Further, we use the notation y_{lm} to denote the y -coordinates of points. Here the first subscript l is the index of the step in the recursive process and subscript m is index of share to which that y -coordinate belongs to. For example, the y -coordinate of share 3 in the 5th recursion is written y_{53} .

The proposed algorithm works as follows - randomly and uniformly choose a number y_{11} and map it as point $(1, y_{11})$. Using $(0, s_1)$ and $(1, y_{11})$ interpolate 1st degree polynomial $p_1(x)$. Sample $p_1(x)$ at two points $y_{21} = p_1(x=2)$ and $y_{22} = p_1(x=3)$. Now map the sampled points as $(1, y_{21})$ and $(2, y_{22})$. Using the next piece as point $(0, s_2)$ and the newly generated points $(1, y_{21})$ and $(2, y_{22})$ interpolate 2nd degree polynomial $p_2(x)$. Evaluate $p_2(x)$ at 3 points $y_{31} = p_2(x=3)$, $y_{32} = p_2(x=4)$, and $y_{33} = p_2(x=5)$. We then use these 3 points as y -coordinates for $x=1, 2, 3$ and along with the

third piece of the message as point $(0, s_3)$ interpolate 3rd degree polynomial $p_3(x)$. We continue this process until we have used all the pieces and reached $(0, s_{k-2})$ and interpolated $(k-2)$ th degree polynomial $p_{k-2}(x)$. We then sample $p_{k-2}(x)$ at $k-1$ points $y_{(k-1)1} = p_{k-2}(k-1)$, $y_{(k-1)2} = p_{k-2}(k)$, $y_{(k-1)3} = p_{k-2}(k+1)$, ..., $y_{(k-1)(k-1)} = p_{k-2}(2k-3)$.

Mapping these $k-1$ samples as points $(1, y_{(k-1)1})$, $(2, y_{(k-1)2})$, ..., $(k-1, y_{(k-1)(k-1)})$ along with $(0, S)$ construct a $(k-1)$ th degree polynomial $p_{k-1}(x)$. We can now sample $p_{k-1}(x)$ at n points such that any k points would reconstruct the secret and the hidden information.

The process of share creation and information hiding is formally described in Algorithm 2.

Algorithm 2 - Dealing Phase

- 1) Consider $k-2$ secrets $s_i \in \mathbb{Z}_p, 1 \leq i \leq (k-2)$.
- 2) Choose prime $p = \max(s_i, S)$, for all $1 \leq i \leq k-2$.
- 3) Randomly and uniformly choose a number $y_{11} \in \mathbb{Z}_p$ and map it as point $(1, y_{11})$.
- 4) Do for $1 \leq i \leq (k-2)$
 - a) Interpolate points $(0, s_i)$ and (j, y_{ij}) , for all $1 \leq j \leq i$ to generate a i th degree polynomial $p_i(x)$.
 - b) Sample the polynomial $p_i(x)$ at $i+1$ points: $y_{(i+1)j} = p_i(j+i)$, for all $1 \leq j \leq (i+1)$.
 - c) Map the $i+1$ points as: $(j, y_{(i+1)j})$, for all $1 \leq j \leq (i+1)$.
- 5) Interpolate points $(0, S)$ and $(j, y_{(k-1)j})$, for all $1 \leq j \leq (k-1)$ to generate $(k-1)$ th degree polynomial $p_{k-1}(x)$.
- 6) Sample $p_{k-1}(x)$ at n points to generate n shares: $(i, p_{k-1}(i))$, for all $k \leq i \leq k+n-1$.

Algorithm 2 - Reconstruction Phase

- 1) Interpolate any k shares to generate $(k-1)$ th degree polynomial $p_{k-1}(x) = S + a_1 x + a_2 x^2 + \dots + a_{k-1} x^{k-1}$.
- 2) Evaluate $S = p_{k-1}(0)$.
- 3) Do for $i = k-2$ down to 1
 - a) Map the coefficients of polynomial $p_i(x)$ as points: (j, a_j) , for all $(i+1) \leq j \leq 2(i+1)$.
 - b) Interpolate (j, a_j) , for all $(i+1) \leq j \leq 2(i+1)$, to generate polynomial $p_i(x)$ of degree i .
 - c) Evaluate $s_i = p_i(0)$.

Security of the proposed method: Algorithm 2 works by repetitive application of Algorithm 1. The first iteration of the algorithm is a direct application of (2,2) Shamir's secret sharing scheme. It uses a polynomial of degree 1 and generates two shares for the first secret s_1 of the message. These two shares may be viewed as random numbers, such that given any number $r \in \mathbb{Z}_p$, $Pr(r = y_{21}) = Pr(r = y_{22}) = \frac{1}{p}$. They are then used to create a quadratic equation along with the second secret s_2 mapped at $x=0$ (the free term of the equation). This quadratic equation is then sampled at 3 points to generate 3 shares of s_2 . These three shares are then used as random points to generate a 4th degree equation and encode s_3 and so on, until we have encoded all the $k-2$ pieces and generated $k-1$ shares. These $k-1$ shares are then used as points along with secret S at $x=0$ to generate a polynomial of degree $k-1$, which can then be sampled at n points to create the final shares. These final shares have the shares of the smaller pieces hidden within them. The security of the protocol is predicated upon the random and uniform choice of the first coefficient y_{11} .

Example. Suppose we want to hide 3 secrets $s_1 = 46$, $s_2 = 69$, and $s_3 = 72$ within the shares of a secret $S=65$. Let $k = 5$ and $n = 7$, i.e. we are to create 7 pieces such that 5 of them must come together to recreate the secret and the hidden message. We execute the algorithm as follows,

- 1) Choose a prime $p = 131$.
- 2) Randomly and uniformly choose a number $y_{11} \in Z_{131}$, say $y_{11} = 102$. Map it as point $(1, 102)$.
- 3) Interpolate $(0, s_1) = (0, 46)$ and $(1, 102)$ to generate $p_1(x) = 56x + 46$.
- 4) Sample $p_1(x)$ at two points $x = 2, 3$: $y_{21} = p_1(2) = 27$ and $y_{22} = p_1(3) = 83$.
- 5) Map these new points as $(1, y_{21}) = (1, 27)$ and $(2, y_{22}) = (2, 83)$.
- 6) Interpolate $(0, s_2) = (0, 69)$, $(1, 27)$ and $(2, 83)$ to generate $p_2(x) = 49x^2 + 40x + 69$.
- 7) Sample $p_2(x)$ at three points $x = 3, 4, 5$: $y_{31} = p_2(3) = 106$, $y_{32} = p_2(4) = 96$ and $y_{33} = p_2(5) = 53$.
- 8) Map the new points as: $(1, y_{31}) = (1, 106)$, $(2, y_{32}) = (2, 96)$ and $(3, y_{33}) = (3, 53)$.
- 9) Interpolate $(0, s_3) = (0, 72)$, $(1, 106)$, $(2, 96)$ and $(3, 53)$ to generate $p_3(x) = 111x^3 + 38x^2 + 16x + 72$.
- 10) Sample $p_3(x)$ at 4 points $x = 4, 5, 6, 7$: $y_{41} = p_3(4) = 119$, $y_{42} = p_3(5) = 43$, $y_{43} = p_3(6) = 98$ and $y_{44} = p_3(7) = 33$.
- 11) Map the new points as $(1, y_{41}) = (1, 119)$, $(2, y_{42}) = (2, 43)$, $(3, y_{43}) = (3, 98)$ and $(4, y_{44}) = (4, 33)$.
- 12) Interpolate $(0, S) = (0, 65)$, $(1, 119)$, $(2, 43)$, $(3, 98)$ and $(4, 33)$ to generate $p_4(x) = 66x^4 + 106x^3 + 72x^2 + 72x + 65$.
- 13) Sample $p_4(x)$ at 7 points $x = 5, 6, 7, 8, 9, 10, 11$ to create 7 shares: $(5, p_4(5)) = (5, 2)$; $(6, p_4(6)) = (6, 40)$; $(7, p_4(7)) = (7, 63)$; $(8, p_4(8)) = (8, 130)$; $(9, p_4(9)) = (9, 50)$; $(10, p_4(10)) = (10, 37)$ and $(11, p_4(11)) = (11, 55)$.

Any five out of the seven shares can be interpolated to regenerate the polynomial $p_4(x)$. This polynomial can then be sampled to obtain $S = p_4(0)$. The y -coordinates of the samples of $p_4(x)$ at points $x = 1, 2, 3, 4$ can be mapped as points at $x = 4, 5, 6, 7$ and then interpolated to reconstruct $p_3(x)$, which can be sampled at $x = 0$ to obtain $s_3 = p_3(0)$. Polynomial $p_3(x)$ can be sampled at $x = 1, 2, 3$ to obtain y -coordinates and map them at $x = 3, 4, 5$. Interpolating these new points we obtain $p_2(x)$ and so on. The pieces of the hidden secrets are retrieved in the reverse order.

III. CONCLUSIONS

We have proposed a recursive techniques to hide additional information within the shares of Shamir's secret sharing schemes. This hidden information may be used for validation of shares at the time of secret reconstruction. Further it may be looked upon as a way to share large secrets by dividing the secret in smaller pieces and recursively hiding them in the shares.

Such a scheme is useful for secure transmission of information over parallel channels. Suppose the transmitter and receiver share secret identifications. The transmitter can then divide the identification into pieces and recursively encode it

into the shares of the message to be sent over parallel lines. Transmission of shares over parallel channels provided implicit security and reliability. Further, the scheme may be used for information dispersal in storage networks.

Future Work. An implementation of an information dispersal scheme in which users store their data on the Web on different servers by creating shares of the data using the proposed scheme. This implicitly prevents any one (compromised) server from having access all the user data [19]. Such an idea would be useful in cloud computing paradigm. Issues regarding addressing of data shares on the network need to be investigated. Further, use of the proposed scheme in existing protocols such a Chord protocol and FreeNets would be of interest.

REFERENCES

- [1] G. R. Ganger and et. al., "Survivable storage systems," in *In DARPA Information Survivability Conference and Exposition, IEEE*. IEEE Computer Society, 2001, pp. 184–195.
- [2] A. Iyengar, R. Cahn, J. A. Garay, and C. Jutla, "Design and implementation of a secure distributed data repository," in *In Proc. of the 14th IFIP Internat. Information Security Conf*, 1998, pp. 123–135.
- [3] P. Rogaway and M. Bellare, "Robust computational secret sharing and a unified account of classical secret-sharing goals," in *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2007, pp. 172–184.
- [4] V. Vinod, A. Narayanan, K. Srinathan, C. P. Rangan, and K. Kim, "On the power of computational secret sharing," *Indocrypt 2003*, vol. 2904, pp. 265–293, 2003.
- [5] P. Beguin and A. Cresti, "General short computational secret sharing schemes," in *Advances in Cryptology EUROCRYPT 95, LNCS vol. 921*. Springer, 1995, pp. 194–208.
- [6] M. O. Rabin, "Efficient dispersal of information for security, load balancing and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.
- [7] J. Garay, R. Gennaro, C. Jutla, and T. Rabin, "Secure distributed storage and retrieval," *Theoretical Computer Science*, pp. 275–289, 1997.
- [8] H. Krawczyk, "Secret sharing made short," *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pp. 136–146, 1994.
- [9] —, "Distributed fingerprints and secure information dispersal," in *PODC 1993*. ACM Press, 1993, pp. 207–218.
- [10] L. Harn, "Efficient sharing (broadcasting) of multiple secrets," *IEE Proceedings - Computers and Digital Techniques*, vol. 142, no. 3, pp. 237–240, May 1995.
- [11] H.-Y. Chien, J.-K. Jan, and Y.-M. Tseng, "A practical (t,n) multi-secret sharing scheme," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 83, no. 12, pp. 2762–2765, 2000.
- [12] L.-J. Pang and Y.-M. Wang, "A new (t,n) multi-secret sharing scheme based on shamir's secret sharing," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 840 – 848, 2005.
- [13] C.-W. Chan and C.-C. Chang, "A scheme for threshold multi-secret sharing," *Applied Mathematics and Computation*, vol. 166, no. 1, pp. 1 – 14, 2005.
- [14] M. Liu, L. Xiao, and Z. Zhang, "Linear multi-secret sharing schemes based on multi-party computation," *Finite Fields and Their Applications*, vol. 12, no. 4, pp. 704 – 713, 2006.
- [15] M. H. Dehkordi and S. Mashhadi, "New efficient and practical verifiable multi-secret sharing schemes," *Information Sciences*, vol. 178, no. 9, pp. 2262 – 2274, 2008.
- [16] M. Gnanaguruparan and S. Kak, "Recursive hiding of secrets in visual cryptography," *Cryptologia*, vol. 26, pp. 68–76, 2002.
- [17] A. Parakh and S. Kak, "A recursive threshold visual cryptography scheme," *Cryptology ePrint Archive, Report 535*, 2008.
- [18] A. Shamir, "How to share a secret," *Communications of ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [19] A. Parakh and S. Kak, "Online data storage using implicit security," *Information Sciences*, vol. 179, no. 19, pp. 3323 – 3331, 2009.