Secret Sharing Made Short

Hugo Krawczyk

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

Abstract. A well-known fact in the theory of secret sharing schemes is that shares must be of length *at least* as the secret itself. However, the proof of this lower bound uses the notion of information theoretic secrecy. A natural (and very practical) question is whether one can do better for secret sharing if the notion of secrecy is *computational*, namely, against resource bounded adversaries. In this note we observe that, indeed, one can do much better in the computational model (which is the one used in most applications).

We present an m-threshold scheme, where m shares recover the secret but $m-1$ shares give no (computational) information on the secret, in which shares corresponding to a secret S are of size $\frac{|S|}{m}$ plus a short piece of information whose length does not depend on the secret size but just in the security parameter. (The bound of $\frac{|S|}{m}$ is clearly optimal if the secret is to be recovered from m shares). Therefore, for moderately large secrets (a confidential file, a long message, a large data base) the savings in space and communication over traditional schemes is remarkable.

The scheme is very simple and combines in a natural way traditional (perfect) secret sharing schemes, encryption, and information dispersal. It is provable secure given a secure (e.g., private key) encryption function.

1 Introduction

Since their invention 15 years ago, secret sharing schemes [16, 2] have been extensively investigated. In particular, much work was done on the required length of the shares relative to the secret size. It is a well known basic fact that shares of a secret have to be at least of the size of the secret itself, and most of the work on share sizes investigates when this lower bound can be achieved or must be exceeded for different kind of schemes. Having shares of the size of the secret is not a serious problem as long as these secrets are short, e.g. a short secret key, as most traditional applications require. However, this effect of information replication among the participants of a distributed environment can be very space and communication inefficient if the secret is a large confidential file, a long message to be transmitted over unreliable links, or a secret data base shared by several servers. Applications like these are becoming more and more necessary.

The mentioned lower bound on the share size is related to the treatment of these secret sharing schemes in the sense of perfect (information theoretic) secrecy. A natural (and practical!) question is what can be done if the secrecy

D.R. Stinson (Ed.): Advances in Cryptology - CRYPTO '93, LNCS 773, pp. 136-146, 1994.

be not perfect but in a computational sense (i.e. against resource-bounded adversaries). [1] Could the shares in this case be made significantly shorter relative to the secret size?

In this note we present a solution to the above question which is surprising in two senses.

- The resultant scheme is extremely space (and communication) efficient. It realizes an m-threshold scheme, where m shares recover the secret but $m-1$ shares give no (computational) information on the secret, in which shares corresponding to a secret S are of size $\frac{|S|}{m}$ plus a short piece of information whose length does not depend on the secret size but just in the security parameter. The bound of $\frac{|S|}{m}$ is clearly optimal if the secret is to be recovered from m shares.
- The resultant solution is strikingly simple. It just combines in a natural way traditional secret sharing schemes, with encryption and information dispersal techniques.

Our scheme is simple, practical and provable secure given a secure private key encryption system (in particular, it just requires the existence of a one-way function). In addition, we present a secret sharing scheme with the same properties as above which is also *robust*, namely, malicious participants cannot prevent the reconstruction of the secret by a legal coalition, even if they return modified shares. (Clearly, the total number of malicious participants must be under some bound). The latter can be achieved by using public-key signatures or, in a much more efficient way, by using the recently introduced ***distributed fingerprints*** [9], together with the above mentioned techniques.

These constructions have many applications, especially in distributed scenarios where secrecy and integrity of information are to be protected. For example, consider a group of five servers sharing a data base of confidential information such that no pair of servers is allowed to learn about the information without the collaboration of a third one. (That is, the system tolerates up to two corrupted servers without compromising the information). Using a regular secret sharing scheme the amount of information stored in each server is equivalent to the whole data base size; in contrast, using our scheme each server keeps one third of the data base size (in other words, the total amount of information in the system has a 66% increase over the data base size in our solution compared to a 400% increase using the regular schemes). Same savings correspond to the amount of communication involved between the servers. Moreover, both storage and communications in this case are secret, and therefore the savings are even more significant.

A particularly interesting application of these space-efficient and robust secret sharing schemes is to the problem of *secure message transmission* defined in

[1] Computational secrecy is in no way a practical limitation. In fact, most implementations of theoretically perfect secret sharing schemes result in actual computational secrecy. This is the case, for example, when shares are encrypted for distribution or when the shares are produced with a pseudorandom generator.

[6]. There, two parties in an (incomplete) network try to communicate a confidential message. Part of the nodes of the network are controlled by an adversary that may want to derive information about the message as well as to corrupt it. The underlying idea in [6] (which investigates this problem in the information theoretic model) is that if n disjoint paths exist between the two parties, such that at most m of them are controlled by an adversary, then a solution to the problem is to decompose the message into n secret shares corresponding to an m-threshold scheme and transmit these shares over the n paths. In this solution, the complexity is given by the cost of computing, transmitting and storing these shares. Clearly, applying our solution significantly reduces this complexity relative to traditional secret sharing schemes.

2 Computational Secret Sharing

An (n, m)-secret sharing scheme is a randomized protocol for the distribution of a secret S among n parties such that the recovery of the secret is possible out of m shares for a fixed value $m, 1 \leq m \leq n$, while $m-1$ shares give no information on the secret S.

In this paper we deal with two different notions of secrecy according to the meaning of "no information" in the above formulation. One is *perfect* secrecy where "no information" is in the information theoretic sense; the other is *computational* secrecy where "no information" means no information that can be efficiently computed. We extend on these notions below.

Therefore, an (n, m)-secret sharing scheme consists of two processes; one the ***distribution*** process, the other the ***reconstruction*** process. The distribution process gets as input the secret S (and the values of n and m) and generates n shares $S_1, S_2, \ldots, S_n$, which are *privately* delivered to the system participants. The reconstruction process reconstructs the correct secret when input with any subset of m shares. Given a particular secret sharing algorithm A we denote by $A(S)$ the n resultant shares $S_1, S_2, \ldots, S_n$. Note that $A(S)$ is a random variable depending on the internal random coins of A.
For simplicity we do not formalize the exact domain of secrets and shares; in general this domain will depend on the specific scheme being used.

The notion of perfect secret sharing is well known and formalized in the literature. The notion of computational secret sharing, although very natural and widely used, is usually implicitly understood and less explicitly formalized. Although such a formalization is not the goal of this note, we outline here the basis for a formal definition. This is required in order to be able to prove that the particular construction we present is *secure*.

We start by outlining the definition of a secure encryption system. This has two reasons, one is that it is in such a secure function that the security of our construction relies; the other is that the definition of computational secret sharing schemes closely follows the definition of secure encryption. Rigorous definitions for secure encryption were first given in [8]. The reader is also referred to [7] for a detailed presentation of these notions, and for the treatment of the security of private key systems (as used in our paper).

The definitions presented here rely on the notion of ***polynomial indistinguishability.***
Roughly speaking, two probability distributions are polynomially indistinguishable if any probabilistic polynomial-time algorithm behaves essentially the same when its input is selected from either of the two distributions. This notion is formalized by means of probabilistic polynomial time *tests* that output 0 or 1 as their guesses for whether the input comes from the first distribution or from the second. The condition for indistinguishability is that any such test will succeed in guessing the correct distribution with probability at most $\frac{1}{2}$ plus a negligible fraction (this probability depends on an equiprobable selection of any of the two distributions, the choice of the input according to the selected distribution and the internal coins of the test).
The formal notion of indistinguishability is an asymptotical one and has to be stated in term of collections of probability distributions indexed, in our case, by the lengths of messages or secrets. Under such formalism, a distinguishing algorithm is one that succeeds in guessing the correct distribution with probability $\frac{1}{2} + l^{-c}$, where l is the distribution index and c a positive constant.

Definition 1. (sketch)

1. Let ENC be an encryption function and M a message in the domain of ENC. Let $\{ENC_K(M)\}_K$ be the space of encryptions of the message M under all possible keys. By $\mathcal{D}_{ENC}(M)$ we denote the probability distribution on $\{ENC_K(M)\}_K$ as induced by the distribution under which keys are selected.
2. A (private-key) *encryption function ENC is secure* if for any pair of messages M' and M'' of the same length, the distributions $\mathcal{D}_{ENC}(M')$ and $\mathcal{D}_{ENC}(M'')$ are polynomially indistinguishable.

This definition, based on the notion of indistinguishability, is equivalent to the (possibly more natural) definition of *semantic security* that states that no information on a message can be derived, in polynomial-time, from seeing its encryption if the key is not known (except for a-priori knowledge on the message). Notice that the above is a weak definition in the sense, for example, that it does not contemplate security against an adversary with additional known-plaintext information. It turns out from our results that even this weak definition suffices for constructing secure and space efficient secret sharing schemes (basically, our scheme uses a "one-time key" for each secret).

Accurate definitions, and a proof of equivalence of the indistinguishability and semantic notions can be found in [7]. (As well as a precise distinction between uniform and non-uniform definitions, an important issue that we overlook here).

We now proceed to define a *computationally secure secret sharing scheme.*

Definition 2. (sketch)

1. Let CSS be an (n,m)-secret sharing scheme ('C' is for computational). For any secret S and for any set of indices $1 \leq i_1 \leq \cdots \leq i_r \leq n$, $1 \leq r \leq n$, let $\mathcal{D}_{CSS}(S, i_1, i_2, \ldots, i_r)$ denote the probability distribution on the set of shares $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$ induced by the output of $CSS(S)$.
2. An (n,m)-secret sharing scheme is *computationally secure* if for any pair of secrets S' and S'' of the same length, and for any set of indices $i_1, i_2, \ldots, i_r$, $r < m$, the distributions $\mathcal{D}_{CSS}(S', i_1, i_2, \ldots, i_r)$ and $\mathcal{D}_{CSS}(S'', i_1, i_2, \ldots, i_r)$ are polynomially indistinguishable.

As in the case of encryption, also computational secret sharing schemes can be defined in the sense of semantic security. The equivalence with the above formulation is proved in a similar way to the encryption case.

A stronger definition can be stated in terms of a dynamic and adaptive adversary that progressively chooses the $m-1$ shares to be revealed to him depending on previously opened shares. Our construction satisfies also such a stronger definition.

Finally, notice that the traditional notion of perfect secret sharing can be defined in an analogous way to Definition 2 by replacing 'polynomially indistinguishable' with 'identical' (or equivalently, by replacing polynomial-time distinguishability tests with computationally unlimited tests).

3 Secret Sharing with Short Shares

In order to achieve a space efficient secret sharing scheme we combine an information dispersal scheme with a secure encryption scheme and a perfect (e.g. Shamir's) secret sharing scheme.

Information Dispersal was introduced by Rabin [15]. It is a scheme intended for the distribution of a piece of information among n active processors, in such a way that the recovery of the information is possible in the presence of m active processors (i.e. out of m fragments), where m and n are parameters satisfying $1 \leq m \leq n$. The scheme assumes that active processors behave honestly, i.e. returned fragments are unmodified. The basic idea is to add to the information, say a file F, some amount of redundancy and then to partition it into n fragments, each transmitted to one of the parties. Reconstruction of F is possible out of m (legitimate) fragments. Remarkably, each distributed fragment is of length $\frac{|F|}{m}$ which is clearly space optimal. Information dispersal schemes can be implemented in a variety of ways, all corresponding to the notion of erasure codes in the theory of error correcting codes (see [15, 13, 9]). We note that basic information dispersal schemes do not deal with malicious parties or with secrecy of information.

Remark: For completeness, we outline the following simple information dispersal scheme based on Reed Solomon erasure codes. The information to be shared is partitioned into m equal parts where each part is viewed as an element over a finite field (e.g. $GF(p)$, for a large enough prime p). These m elements are then viewed as coefficients of a polynomial of degree $m-1$, and the n fragments for distribution are obtained by evaluating this polynomial in n different points. Clearly the whole information can be reconstructed (by interpolation) from any m fragments. We stress that for large files, it is not necessary to work on a huge field but just to view the information as the concatenation of different polynomials. Notice the difference between such a scheme and Shamir's secret sharing in which the information is represented by the free coefficient and not by the whole polynomial (this is essential in Shamir's scheme to provide perfect secrecy but not for information dispersal where secrecy is not a concern).

We proceed to show how to build a space efficient secret sharing scheme. Let n denote the number of parties among which the secret is to be shared. Let m denote the threshold for our scheme, namely, m shares suffice to construct the secret but $m-1$ give no (computational) information on the secret. Let S denote the secret being shared.

We assume a generic information dispersal algorithm that we denote by IDA, and which works for parameters n (number of file fragments) and m (number of required fragments to reconstruct the file). We also assume a secure (length preserving) private key encryption function, denoted ENC, and a perfect (n, m)-secret sharing scheme (e.g. Shamir's) which we denote PSS. The space of secrets in our scheme is the same as the space of messages for the encryption function ENC.

Distribution Scheme:

1. Choose a random encryption key K. Encrypt the secret S using the encryption function ENC under the key K, let $E = ENC_K(S)$.
2. Using IDA partition the encrypted file E into n fragments, $E_1, E_2, \ldots, E_n$.
3. Using PSS generate n shares for the key K, denoted $K_1, K_2, \ldots, K_n$.
4. Send to each participant P_i, $i = 1, 2, \ldots, n$ the share $S_i = (E_i, K_i)$. The portion K_i is privately transmitted to P_i (e.g. using encryption or any other secure way).

Reconstruction Scheme:

1. Collect from m participants P_{i_j}, $j = 1, 2, \ldots, m$ their shares $S_{i_j} = (E_{i_j}, K_{i_j})$.
2. Using IDA reconstruct E out of the collected values E_{i_j}, $j = 1, 2, \ldots, m$.
3. Using PSS recover the key K out of K_{i_j}, $j = 1, 2, \ldots, m$.
4. Decrypt E using K to recover the secret S.

Theorem 3. *The above scheme constitutes a computationally secure (n, m)-secret sharing scheme provided that ENC is a secure encryption function and PSS a perfect secret sharing scheme. Each share S_i is of length $\frac{|S|}{m} + |K|$.*

Proof. The feasibility to reconstruct the encrypted secret E out of the m fragments E_{i_j} is inherited from the properties of the algorithm IDA. Also the reconstruction of the key K out of K_{i_j} is guaranteed by the secret sharing scheme PSS. Knowledge of E and K permits deriving S using the decryption function. The lengths of the shares comes from the lengths of the fragments and shares, respectively, in these schemes. (We assume the shares corresponding to the key K are of the same size as K. This is always possible given that log $n < |K|$, a very reasonable assumption).

As for the secrecy against a coalition of $m - 1$ shares, the intuitive idea is clear. The $m - 1$ fragments corresponding to E give no more information on S than E itself. On the other hand, the $m - 1$ key-shares give no information at all on K, therefore knowing E cannot help to learn about S.

A formal proof of the secrecy uses the following simulation argument.

Assume there exists a pair of secrets S' and S'' and an algorithm A that distinguishes (with significant probability) between the space of shares corresponding to S' and the space corresponding to S''. We construct an algorithm B to break the encryption function ENC in the sense that B can distinguish between the space of encryptions of S' and the space of encryptions of S''. When B is given an encrypted version of S' or of S'', call it E, it applies IDA on E to generate n fragments $E_1, E_2, \ldots, E_n$, produces at random $m - 1$ shares according to the distribution of shares[2] (the key sharing is perfect!), and gives the fragments and "shares" as input to A. Now A outputs its guess for whether the secret corresponds to S' or S''; B outputs the same guess. Since A guesses correctly with significant probability over $1/2$ then B succeeds in its distinction with same probability. Therefore, B breaks the encryption function in the sense of indistinguishability.

□

We stress that the simplicity of the above proof has an important "practical" aspect. It permits a clear evaluation of the security of the scheme relative to the underlying encryption function even when this encryption function is "practically secure" (e.g. DES), rather than formally secure. Finally, we note that the issue of implementation of the "private channel" necessary between the dealer of the secret and its recipients is orthogonal to the aspect treated here. On the other hand, an insecure implementation of that channel compromises the security of the whole scheme. Using a secure encryption function (private or public key) for these channels the security of the whole system is easily provable.

Remark: Secret sharing schemes with shares shorter than the secret itself were also investigated under the information theoretic model. See, for example, [3] for a description of the so called *ramp schemes*. These schemes give up the perfect uncertainty on the secret provided by perfect secret sharing schemes in order to reduce the length of shares. Unfortunately, the security of these schemes is questionable (and insufficient) in many applications. Although one can show that the exact value of the secret cannot be learned as long as the number of shares revealed is under some threshold, there is a leakage of information with each opened share (below that threshold). The amount of leaked information is easy to measure but not the *significance* of this information, or the hardness of learning it. In other words, the approach is a quantitative one and not semantic as required in cryptography. As a simple example, an attacker to these schemes

[2] This distribution is polynomial-time samplable, e.g. use the PSS algorithm on a randomly chosen secret.

can *efficiently* discard a particular value (or even a set of values) for the secret just after seeing a number of shares much below the reconstruction threshold. In our scheme, on the contrary, if one uses a semantically secure encryption function this (or any other useful) information cannot be efficiently learned even from $m - 1$ shares.

4 Robust Secret Sharing

The basic secret sharing scheme as introduced in section 3 assumes that share holders return correct shares. In many applications this assumption is too strong. This scenario, in which some shares can be (maliciously) corrupted, was investigated in many works (e.g. [10, 14, 17, 4]). A *robust secret sharing scheme* is a secret sharing scheme that can correctly recover the secret even in the presence of a (bounded) number of corrupted shares, while keeping the secrecy requirement.

In this case, in addition to the threshold parameter m, a bound t on the number of malicious parties in the system is to be specified. It is necessary that $t < m$ (a coalition of only malicious parties cannot reconstruct the secret), and $m \leq n - t$ (there are enough good parties to reconstruct the secret). This two relations imply that $2t < n$, i.e. a majority of honest parties is required. Clearly, it is not possible anymore to require that *any* subset of m parties can recover the secret (since part of them can be faulty). Instead we require that any coalition *containing* m honest parties (i.e. from any subset of shares containing at least m correct shares) can reconstruct the secret.

Our goal in this section is to present a robust secret sharing scheme which preserves the space efficiency of the construction described in section 3.

The first solution to the problem of designing robust secret sharing schemes was presented by McElice and Sarwate [10] where error correcting code techniques are used to enhance the original Shamir's scheme against share corruption. It tolerates up to $n/3$ cheaters and the security is unconditional. That is, secrecy is perfect, recovery is guaranteed (with probability 1), and no computational assumptions are done. Their solution cannot be applied to our needs since it requires shares (at least) as long as the secret itself. The same (space) drawback exists in all other solutions designed against computationally unlimited adversaries. A different approach is used by Rabin in [14], where shares are fingerprinted in order to detect possible alterations. Since fingerprints intended to work against resource bounded adversaries can be small and unrelated to the information length, this approach can be used to keep the space efficiency of our construction. Rabin's solution uses public key signatures for fingerprinting. (Notice that fingerprints based on private keys are unsuitable since different – mutually suspicious – parties need to verify the shares).

Therefore, our construction of Section 3 is modified such that at time of share distribution both the fragments E_i corresponding to the encrypted secret E as well as the key shares K_i are signed (using the private signing key of the dealer). When the secret is reconstructed, the public verification key for that dealer is used to verify the correctness of the shares. The total amount of information added to each share depends only on the security parameter and not on the shares or secret themselves. Therefore, the efficiency of our scheme is preserved.

This solution, although space efficient, requires the implementation of a public key system to support public signatures. This has a significant cost in administration, key management and computation. In addition, it requires to know the identity of the dealer that generated the secret (which is not always desirable) and a time-stamp mechanism to avoid replay attacks. (Notice that in our basic solution no need for a public-key system exists).

The above drawbacks related to the use of public-key signatures are overcome using the recently introduced ***distributed fingerprints*** [9] that permit fingerprinting the information through a method that requires no public key system, uses no secret keys at all, is time and space efficient, does not require the signer's identity or time-stamp techniques. It just requires a global public one-way hash function and the existence of a majority of honest parties. The latter condition is also part of the requirements for any robust secret sharing scheme, and therefore does not constitute a limitation here. Moreover, the distributed fingerprint scheme uses distribution among the parties in the system for fingerprint protection, which fits naturally in the secret sharing model. We refer to [9] for details on these fingerprints.

5 Further Work

Our results have many immediate applications (examples appear in the introduction). We expect also less immediate applications to emerge in the future.

In addition, many of the questions investigated for traditional secret sharing schemes are relevant to space-efficient computational secret sharing schemes. We mention here two questions which seem particularly attractive.

In this paper we have dealt with space efficient threshold schemes. More general schemes classified according to their *access structures* (cf. [1]) are investigated in the literature, mostly in the context of perfect secrecy. A natural question is whether the space efficiency can be carried over more general access structures than just threshold schemes. Our scheme can be easily extended to deal with some of these structures but it is not clear how general these structures can be. Since in our approach we apply regular secret sharing schemes to the sharing of the encryption key, then this part of the protocol can be treated as for traditional secret sharing. In this sense the question reduces to deal with access structures for information dispersal. The interested reader is referred to [12] that deals with information dispersal over arbitrary graphs.

Another question is whether *verifiable secret sharing* can be done in a space-efficient way. While robust secret sharing deals with potentially corrupted share holders, verifiable secret sharing deals also with corrupted dealers of the secret that can distribute inconsistent shares in order to prevent legal coalitions of reconstructing the secret (cf. [5]). We mention here, very briefly, the strong relation between space efficient verifiable secret sharing and fair cryptography [11]. A more desirable approach than sharing the keys used to encrypt information with escrow agencies, is to share with them each individual message. (Here 'to share' is in a sense analogous to that of 'secret sharing', namely, the information can be acceded only if a number of escrow agencies collaborate to do that). The drawback with sharing the key is that once a key is "opened" all messages (past and future) encrypted with that key can be open. Sharing individual messages solves this problem, but it is impractical to realize it through regular secret sharing schemes which require the replication of the amount of information for encryption, transmission and storage. In this sense the need for short shares is clear. On the other hand, verifiability is also necessary, otherwise the escrow agencies can be given shares that do not reconstruct the message. Further elaboration of this application is beyond the scope of this paper.

Acknowledgement

I thank Mihir Bellare, Oded Goldreich and Moti Yung for very helpful discussions.

References

1. Benaloh, J. and Leichter J., "Generalized secret sharing and monotone functions", Proc. Crypto '88, pp. 27-35.
2. Blakley, G.R., "Safeguarding Cryptographic Keys", *Proc. AFIPS 1979 National Computer Conference*, New York, Vol. 48, 1979, pp. 313-317.
3. Blakley, G.R., and Meadows C. "Security of Ramp Schemes", in *Lecture Notes in Computer Science 196; Advances in Cryptology: Proc. Crypto '84*, Springer-Verlag, 1985, pp.242-268.
4. Brickel, E.F., and Stinson, D.R., "The Detection of Cheaters in Threshold Schemes", in *Lecture Notes in Computer Science 403; Advances in Cryptology: Proc. Crypto '88*, Springer-Verlag, 1990, pp.564-577.
5. Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. 26th FOCS*, 1985, pp. 383-395.
6. Dolev, D., Dwork, C., Waarts, O., and Yung, M., "Perfectly Secure Message Transmission", *Proc. 31st IEEE Symp. on Foundations of Computer Science*, 1990, pp. 36-45.
7. Goldreich, O., "A Uniform-Complexity Treatment of Encryption and Zero-Knowlege", *Jour. of Cryptology*, Vol. 6, No. 1, 1993, pp.21-53.
8. Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.
9. Krawczyk, H., "Distributed Fingerprints and Secure Information Dispersal", *Proc. of 12th. PODC*, pp. 207-218, 1993.
10. McElice R.J., and Sarwate, D.V., "On Sharing Secrets and Reed-Solomon Codes", *Comm. ACM*, Vol. 24, No. 9, 1978, pp. 583-584.
11. Micali, S., "Fair Public-Key Cryptosystems", Crypto '92.
12. Naor, M., and Roth, R.M., "Optimal File Sharing in Distributed Networks", *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, 1991, pp. 515-525.

13. Preparata, F.P., "Holographic Dispersal and Recovery of Information", *IEEE Trans. on Information Theory*, IT-35, No. 5, 1989, pp. 1123-1124.
14. Rabin, M.O., "Randomized Byzantine Agreement", *24th FOCS*, pp. 403-409, 1983.
15. Rabin, M.O., "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", *Jour. of ACM*, Vol. 36, No. 2, 1989, pp. 335-348.
16. Shamir, A., "How to Share a Secret", *Comm. ACM*, Vol. 22, No. 11, 1979, pp. 612-613.
17. Tompa, M. and H. Woll, "How to share a secret with cheaters", *Journal of Cryptology*, Vol 1, 1988, pp 133-138.