Distributed Steganographic Data Storage with Verifiable Dynamic Reconfiguration

Gevorg Margarov
State Engineering University of Armenia (Polytechnic)
Yerevan, Armenia
e-mail: gmargarov@gmail.com

Vahan Markarov
State Engineering University of Armenia (Polytechnic)
Yerevan, Armenia
e-mail: vmarkarov@yahoo.com

Samvel Soghomonyan
State Engineering University of Armenia (Polytechnic)
Yerevan, Armenia
e-mail: sogsam@seua.am

ABSTRACT
This paper is devoted to creation of high capacity hidden data storage distributed on the Internet. Offered new approach to construction of such systems is based on steganographic content sharing, controllable verification and redistribution of shares. The storage is designed thus to defend against three basic types of the adversary: passive, active and dynamic. The developed algorithm uses threshold sharing schemes and incorporates a verification capability to support redistribution between arbitrary sets of carrier files. It is proved that an adversary cannot combine old shares and new shares to reconstruct the steganographic content. In contrast to similar the offered algorithm considers specific features of steganography and can accommodate dynamic carrier files group membership changes.

Keywords
Steganography, verifiable reconfiguration, secret sharing, distributed storage.

1. INTRODUCTION
The Internet can be considered as the most suitable environment for the latent storage and multi-user access to big volume of data, using a set of carrier files located on various sites [1]. The primary goal of such storage is to preserve the long-term flexible availability and confidentiality of data in the face of carrier files failures and compromises including as result of attacks. Thus specificity of attacks on steganographic systems is that not only the hidden data can be detected, but also carrier files can be changed, damaged or even removed by attacker. Another goal is to adapt to the addition or removal of carrier files. The purpose of this paper is to outline the design for distributed steganographic data storage that meets those goals, and present the method of reconfiguration that is a key component of the storage.

It is usually possible to envision such storage to store data that is infrequently accessed, but which must remain available and confidential for long periods of time. In this case it is possible to trade off longer storage and retrieval latencies in return for stronger availability and confidentiality guarantees. Such nature of the storage allows use relatively heavyweight schemes for distributing the secret data to carrier files. For instance the steganographic content (secret data) can be distributed among N carrier files on base of Shamir's scheme [2] and it is required M (where $M \le N$) shares to reconstruct the secret data. At the same time obviously an adversary must detect at least M carrier files and get from them hidden shares to uncover the latent secret. From the other hand an adversary can detect or damage at least $N - M + 1$ carrier files to make storage disabled [3]. In other words it is possible to consider two versions of compromising distributed steganographic data storage and such approach introduces a degree of fault-tolerance and workability of storage can remain even if $N - M$ carrier files fail. Of course it is necessary to believe that there is enough diversity among the carrier files such that common security flaws and failure modes can be ruled out.

2. BASIC ATTACKS AND DEFENCE AGAINST THEM
The storage is designed thus to defend against three basic types of the adversary:

- passive,
- active,
- dynamic.

A passive adversary may read the carrier files but not modify them or cause the behavior of a carrier to damage or remove. To defend against passive adversary the periodic refreshments of the shares can be performed [3].

An active adversary corrupts data or carrier files, and may alter some of them. To defend against active adversaries, secret sharing must be verifiable as well as in case of cryptography [4] users should be able to verify the correctness of dynamic reconfiguration execution, since an active attacker can interfere in this process.

A dynamic (or mobile) adversary compromises carrier files progressively, and left unchecked will eventually compromise enough carrier files to compromise or make storage disabled. To counteract dynamic adversary the full refreshment of the steganographic data should be executed periodically [3]. This scheme assumes a storage model of temporary compromise (i.e., compromised carrier files can be restored to a clean state by full refreshment), and that the adversary compromises at most $N - M$ carrier files simultaneously prior to refreshment.

The desire to defend against passive, active, and dynamic adversaries require a general sharing scheme in which the steganographic content can be redistributed dynamically to a new set of carrier files after the initial sharing. The new carrier files may form a new access structure, i.e., they may or may not overlap with the old carrier files, and they may or may not keep a proportion between N and M values.

A trivial, but insecure, solution for recovering from lost or compromised carrier files is simply to restore the original steganographic content by the storage on the base of shares,

extracted from M undetected carriers, and again distributes the secret among new set of N carrier files. This solution suffers from an obvious weakness – if adversary can interfere with refreshment process the storage will appear completely compromised and confidential data will be lost irrevocably.

More preferable can be a solution that does not involve the restoration of the steganographic content, in which the work of redistribution is performed by the remaining (non-compromised) carrier files. The similar approach exists for cryptographic systems [5], but specificity of steganography is that in this case the most essential are the attacks connected with disclosing of very fact of hidden data existence. Presented method of verifiable dynamic reconfiguration considers noted specificity of steganography and generally replaces access structure $(M,\ N)$ with new arbitrary structure $(M',\ N')$. Thus essential difference is that the capability to verify validity of the new shares (i.e., that they can be used together with old noncompromised shares to reconstruct the original steganographic content). It is necessary to stress that the ability to perform verification is essential for distributed steganographic systems where having some compromised carrier files is the common case rather than the exception.

3. DATA STORAGE ARCHITECTURE

The high-level view of distributed steganographic data storage architecture is shown on Figure 1.

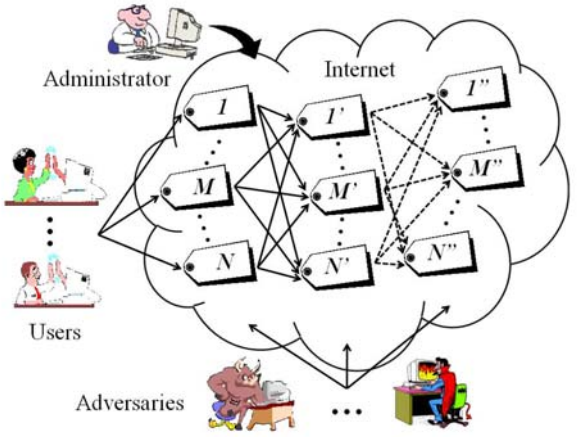

Figure 1. The high-level view of distributed steganographic data storage architecture

Architecture consists of four main components: administrator, users, adversaries and a group of steganographic containers (carrier files), distributed on the Internet. Users perform the initial distribution and final reconstruction of files (i.e., secrets), and are considered trusted entities. Adversaries try to find out the fact of existence of steganographic data storage and if they are successful, to uncover the latent secret and to prevent normal functioning of storage (make it disabled).

Administrator manages placing of shares on set of carrier files $(1\ldots\ M\ldots\ N)$ and performs redistribution. Though the number of carrier files that implement the data storage may be very large, it is supposed that the number of carrier files N that store shares for a particular file (or set of files) is relatively small. Thus administrator should provide private point-to-point links between users and corresponding carrier files. Administrator requires mechanisms to keep track of the members of the active group of carrier files, and to determine when carrier files have joined or left (intentionally or through failure) the group. When administrator detects changes in the active group membership, it redistributes the shares to the new group of carrier files $(1'\ldots\ M'\ldots\ N')$. Administrator may perform redistribution an arbitrary number of times $(1''\ldots\ M''\ldots\ N'')$ prior to reconstruction. It is supposed, that the rate of change of membership is low compared to the rate at which users contact the group for I/O operations.

Users require a mechanism to locate the active group of carrier files for I/O operations. Users are not part of the group of carrier files (in contrast to peer-to-peer storage systems such as OceanStore [6]), and thus cannot rely on the group membership protocol used by the administrator. A simple approach would be for the user to contact a central directory that replies with the list of carrier files; administrator would update the directory after a change in group membership. Of course, the central directory is an obvious point of vulnerability in an otherwise decentralized architecture. A more robust approach is for the user to contact a replicated directory service that uses agreement protocols to ensure consistent and valid updates to the list of carrier files (such as in Farsite [7]).

Users also require a heuristic to select the threshold value M for given N carrier files. User requires M non-faulty carrier files to reconstruct the secret, and can tolerate at most $M-1$ faulty carrier files. Thus, it is necessary, that $M+M-1\le N$, or $M\le(N-1)/2$. To store a file in the steganographic data storage, a user locates the active group of N carrier files and selects the $(M,\ N)$ access structure to use. It then distributes N shares of the secret to the carrier files and a witness to the secret to administrator (described in Section 4.3). When administrator detect that a carrier file left the group it have to join to the group another carrier file and redistribute shares of the secret to the new group. Administrator use the same heuristic as the users to select the new select the threshold value M' for N' carrier files of the new group. Administrator may redistribute the file an arbitrary number of times, not informing corresponding users. Finally, when a user needs to reconstruct the secret, it locates the active group of carrier files, which may differ from the group to which it distributed shares initially. The user then retrieves at least M' shares and reconstructs the secret.

4. BASICS OF RECONFIGURATION

4.1. Steganographic content sharing scheme

Shamir's $(M,\ N)$ threshold sharing scheme is based on polynomial interpolation [2]. To distribute secret K to the access structure $(M,\ N)$ it is necessary to select an $M-1$ degree polynomial $A(x)$ with constant term K and random coefficients $A_1\ \ldots\ A_{M-1}$ and use it to generate shares S_i for hiding in set of carrier files α $(|\alpha|=N)$:

$$S_i = K + A_1 i + A_2 i^2 + \ \ldots\ + A_{M-1} i^{M-1} \qquad (1)$$

To reconstruct K, it is necessary to retrieve M shares S_i form set of carrier files β ($|\beta| = M$), and use Lagrange interpolation:

$$K = \sum_{i \in \beta} S_i \prod_{j \in \beta, j \neq i} \frac{j}{j - i} \quad (2)$$

Thus it is obvious that $\beta \in \alpha$, otherwise it is impossible to restore the shared secret correctly.

4.2. Redistribution of shares

For the redistribution of shares of secret K from access structure (M, N) to access structure (M', N') without requiring the intermediate reconstruction of the secret using the authorized subset β for above considered sharing scheme by analogy with [5] it is necessary for each $i \in \beta$, to compute the subshares $\hat{S}_{ij}$ of S_i on base of the polynomial $A'_i(j) = S_i + A'_{i1} j + A'_{i2} j^2 + \ldots + A'_{i(M'-1)} j^{M'-1}$ and send $\hat{S}_{ij}$ to the corresponding $j \in \alpha'$. Later for each $j \in \alpha'$, generate a new share S'_j by Lagrange interpolation:

$$S'_j = \sum_{i \in \beta} \hat{S}_{ij} \prod_{x \in \beta, x \neq i} \frac{x}{x - i} \quad (3)$$

4.3. Verifiable secret sharing

The application of verification approach to above considered sharing scheme by analogy with [4] takes advantage of the homomorphic properties of exponentiation and the assumption that the computation of discrete logs in a finite field is intractable.

Suppose we have fields Z_p and Z_r, such that p and r are prime and $r = pq + 1$ (where q is a non-negative integer), and suppose we have an element $g \in Z_r$ of order p. Then, suppose we use above considered sharing scheme with polynomial $A(x)$ to distribute a secret $K \in Z_p$ to the access structure (M, N). In addition to sending the shares $S_i \in Z_p$ to i carrier files, we broadcast witnesses to K and the coefficients $A_1 \ldots A_{M-1}$ of $A(x)$ in the form of g^K and $g^{A_1} \ldots g^{A_{M-1}}$. For each carrier file it is possible to verify that S_i is a valid share of K by equation:

$$g^{S_i} = g^K (g^{A_1})^i (g^{A_2})^{i^2} \ldots (g^{A_{M-1}})^{i^{M-1}} \quad (4)$$

which is the exponentiation of $A(x)$. Since it is assumed that the computation of discrete logs is intractable, no-one can calculate K or $A_1 \ldots A_{M-1}$ from the broadcast of the witnesses.

5. VERIFIABLE CONTENT REDISTRIBUTION ALGORITHM

Suggested verifiable steganographic content (secret data) redistribution algorithm is based on above considered three basic approaches – data sharing, redistribution and verification. The algorithm takes as input shares of a steganographic content distributed to the access structure (M, N), and outputs shares redistributed to the access structure (M', N'). It is assumed that the computation of discrete logs in a finite field is intractable, and that there exist reliable broadcast and private channels among all users and administrator. It is also assumed that there are at least M non-faulty old carrier files, that there are at most $M - 1$ faulty old carrier files, and that there are N' non-faulty new carrier files.

The initial distribution of steganographic content proceeds as in verifiable secret sharing scheme (Section 4.3). The user distributes the secret K to N carrier files so, that each i share is defined by the polynomial $A(i)$. The user also broadcasts g^K and $g^{A_1} \ldots g^{A_{M-1}}$, which each i uses in Equation (4) to verify the validity of S_i. If Equation (5) holds, administrator stores S_i in carrier file i and corresponding witnesses in special table.

Reconfiguration of the distributed steganographic data storage proceeds as in inedistribution of shares scheme (Section 4.2). For each carrier file i from an authorized subset β administrator uses Shamir's scheme (with the polynomial $A'_i(j)$) to distribute subshares $\hat{S}_{ij}$ of its share S_i to access structure (M', N'). Administrator may redistribute K an arbitrary number of times before user reconstruct it.

For the new carrier files to verify that their shares of the secret are valid after redistribution, it is required that two conditions hold. When for all $i \in \beta$ S_i redistributed to each $j \in \alpha'$, all S'_j are valid shares of K if hold conditions of Equation (2) and Equation (3) for β' of access structure (M', N'). That is:

$$S_i = \sum_{j \in \beta'} \hat{S}_{ij} \prod_{x \in \beta', x \neq j} \frac{x}{x - j} \quad (5)$$

For the new shares S'_j the condition of validity follows from Equation (3) for a secret K distributed to access structure (M', N'). That is:

$$K = \sum_{j \in \beta'} S'_j \prod_{x \in \beta', x \neq j} \frac{x}{x - j} \quad (6)$$

In this case administrator has to contact corresponding users to update table of witnesses. It is necessary to allow users to verify of shares from the new carrier files. In table of witnesses for each $i \in \beta$ must therefore be stored g^K. Each user can receives S_i from each carrier file to verify that subshares holds the conditions and can receives witnesses from the table (including g^K) to verify that S_i is a valid share of K by:

$$g^K = \prod_{i \in \beta} g^{b_i S_i}, \text{ where } b_i = \prod_{x \in \beta, x \neq i} \frac{x}{x - i} \qquad (7)$$

Equation (7) follows from Equation (2) and the homomorphic properties of exponentiation. Since it is assumed that the computation of discrete logs is intractable, no-one can calculate K from the broadcast of g^K.

6. SUMMARY

The offered new approach to construction of steganographic systems allows creating high capacity hidden data storage distributed on the Internet. It is based on simultaneous use of three basic methods – steganographic content sharing, controllable verification and redistribution of shares. The nature of such systems itself calls for heavyweight protection mechanisms to ensure the long-term availability and confidentiality of stored data. Additionally, it is necessary to account for the addition and removal of steganographic carrier files within the lifetime of the data. The storage is designed thus to defend against three basic types of the adversary: passive, active and dynamic. The developed algorithm uses threshold sharing schemes and incorporates a verification capability to support redistribution between arbitrary sets of carrier files.

The analysis of vulnerabilitys of similar cryptographic algorithms has allowed showing that two conditions (shares validity and subshares validity) are sufficient to guarantee that new carrier files have valid shares after redistribution. It is also proved that an adversary cannot combine old shares and new shares to reconstruct the steganographic content, provided that the adversary has less than M old shares and M' new shares. The developed redistribution algorithm can tolerate up to $M-1$ faulty old carrier files (provided that there are at least M honest members). It is pointed out that the identification and replacement of faulty members of carrier files active group is not immediately possible if the new members must rely on the old carrier files to distribute verification information. It can be shown, that at worst case $\binom{N}{M} - \binom{N-M+1}{M}$ restarts are required to eliminate faulty carrier files and complete the algorithm.

In contrast to similar the offered algorithm considers specific features of steganography and can accommodate dynamic carrier files group membership changes. The important feature of algorithm is possibility to guard against mobile (dynamic) adversaries with permanent compromise. That is, it is possible to deal with compromise that cannot be recovered with a reboot operation. Of course it is still required that at any given point of time, the number of faulty carrier files in the active group of containers is less than the threshold value.

For the further development of the offered approach to construction of high capacity distributed steganographic systems and its finishing to practical realisation it is necessary to study in details the nature of steganographic carrier files failures in a consequence to the "natural" reasons and as a result of adversary's purposeful influence. Besides it would be useful to have a real estimation of complexity of algorithm for the conditions representing practical interest. Also it would be necessary to present the strict proof of a correctness and security of the offered approach.

REFERENCES

[1] G. Margarov, "Data Hiding on the Internet: Steganalysis against Steganography", *Proceedings of Advanced Research Workshop "Old threats, new channels – the Internet as a tool for terrorists"*, Berlin, 2008.

[2] A. Shamir, "How to Share a Secret", *Communications of the ACM*, Volume 22, Issue 11, 612-613, 1979.

[3] G. Margarov, V. Markarov, A. Khachaturov, "Steganographic system with dynamically reconfigurable structure", *Proceedings of the 2009 International Conference on Security & Management, SAM'09*, Volume 1, Las Vegas, 43-45, CSREA Press, 2009.

[4] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing", *Proceedings of the 28th IEEE Ann. Symp. on Foundations of Computer Science*, 427–437, 1987.

[5] Y. Desmedt, S. Jajodia, "Redistributing secret shares to new access structures and its applications", *Technical Report ISSE TR-97-01, George Mason University*, Fairfax, VA, 1997.

[6] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao. "OceanStore: An architecture for global-state persistent storage", *Proceedings of ASPLOS IX, the Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp 190–201, Nov. 2000.

[7] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment", *Proceedings of the 5th Symp. on Operating Systems Design and Implementation*. Dec. 2002.