Innovations Syst Softw Eng (xxxx) x:xx–xx
DOI

ORIGINAL PAPER

On the Use of Secret Sharing as a Secure Multi-use Pad

Neil Buckley • Atulya K. Nagar • S. Arumugam

Abstract Secret sharing (SS) is a cryptographic method proposed independently by Adi Shamir and George Blakley in 1979 to encode the keys of public-key cryptography by splitting them into maximally entropic shares that are distributed to participants, only revealing the secret when combined. Each new sharing instance, even of the same key, produces a different set of shares to distribute anew. This paper investigates SS as an independent cipher to secure confidential messages between a limited set of trusted participants by eliminating the need to redistribute shares. A participant's master share is permanently fixed and unlimited temporary shares are created and combined with it to reveal new messages. Security is argued against specific and general attacks.

Keywords secret sharing · cryptography · cybersecurity

This research was funded by Higher Education Institutional Research (HEIR) and Liverpool Hope University. The work is patent pending, application number GB1504243.5. The authors declare no conflict of interest in this research.

N. Buckley
Liverpool Hope University, England
e-mail: 08008783@hope.ac.uk
Corresponding author, tel.: 0151-291-3143

A.K. Nagar
Liverpool Hope University, England
e-mail: nagara@hope.ac.uk

S. Arumugam
Kalasalingam Unversity, India
email: s.arumugam.klu@gmail.com

1 Introduction

Secret Sharing (SS) was developed by Shamir[20] and Blakley[5] as a solution to the secure distribution of keys in public-key cryptography. In Shamir's Secret Sharing (SSS), there are n shares, any k of which can combine to reveal the secret. Each share is maximally entropic, revealing no information about the secret, even with infinite computation.

As proved in [20], it is information-theoretically secure, not relying on computational problems such as prime factorisation, so various uses have been proposed and developed, such as secure multi-party computation (MPC). However, each encoding produces a set of new shares to be distributed to participants via potentially non-secure channels, and it is possible for a hacker to intercept sufficient shares. This makes SSS in its original form impractical for use, by itself, in messaging systems. Such systems include email, SMS or any textual messaging service. Indeed, emails often lack security [16], necessitating extra encryption (commonly RSA [19]), and with the introduction of quantum computation such as the D-Wave 2 [7], stronger security is desired.

This paper proposes SSS to secure messages between a limited set of trusted users. The need to distribute all shares for every message is eliminated with a collection of fixed master shares held by all users. These are distributed only once, after they have been generated (here analogous to key generation). Each transmission creates a temporary "transient" share, sent over the network and combined with the recipient's master share to reveal the message. As such, although only (2, 2)-SSS (i.e. where a threshold of two shares are required out of a total of two shares) is used in this paper, the advantage of SSS is its ability to form access structures, potentially involving multiple users in a secure transmission. The contributions of this paper are as follows:

- Equations are derived from the standard SSS model for the generation of transient share elements using master share elements and message characters.
- Algorithms are proposed for the creation of master and transient shares that correctly combine to reveal any required confidential message, as well as to decode it on receipt.
- Multi-use master shares are made possible, with statistical linguistic analysis prevented with message and share obfuscation.
- Preliminary revocation and addition methods are suggested.

The remainder of this paper is organised as follows: Section 2 reviews prior secret sharing literature. Section 3 details Shamir's secret sharing. Section 4 proposes secure messaging, including key generation in 4.1, encoding in 4.2 and decoding in 4.3. Section 5 demonstrates secure transmission of messages using these algorithms and discusses various attacks. Finally, Section 6 concludes the study and suggests further research directions.

2 Related Work

In 1979, two methods for the secure secret sharing of cryptographic keys were devised independently in [20] and [5]. In both, the secret can be in character string format, which is converted to numeric format using ASCII values. In Blakley's method, each share is an $(n-1)$-hyperplane with the secret encoded into the unique coordinate in n-dimensional space at which planes intersect.

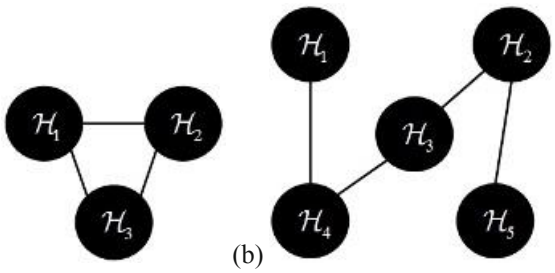
Shamir's SS reduces space complexity versus Blakley's method (since each share in the latter must be at least k times larger than the secret) by encoding the secret into the constant terms of k randomly selected coordinates of an $(n-1)$-degree polynomial. A line, for example, encodes a (2, 2)-SS scheme, since two points are required to reconstruct it and derive its constant term. It therefore benefits from perfect information-theoretic security, as fewer than k points on a $(k-1)$-degree curve can be coordinates on any of an infinitude of $(k-1)$-degree polynomials (assuming real-valued coefficients).

Prior practical application of SSS has been in the concealment of graphical information, whether it is personal bitmap images while in transit via a network, or highly confidential imagery, such as medical images and biometric data. In [21], SSS is used to bring about threshold access structures for sharing graphical medical data, such that shares are held by a large number of professionals. Ref. [1] similarly proposes a method for the concealment of biometrics.

SSS is not the only form of secret sharing applicable to image data. The most well-known method is visual cryptography, developed by Naor and Shamir [18] to build shares that can be printed onto transparent sheets and physically stacked to visually reveal the secret. Although less computationally costly than SSS, which does not permit visual decoding, reconstruction is extremely lossy. Similarly, the random grids method of [11], improved in [22], results in the same kind of shares but benefits from smaller shares and lack of complicated rulebooks.

Circles, as opposed to polynomials, are used in [10] to bring about (3, n)-SS, but only allow threshold access structures, i.e. those representable by fully connected (hyper)graphs, are allowed (as in Figure 1a). The Chinese Remainder Theorem (CRT) can be used to bring about general access, and was first proposed in [3], whose work has since become known as the Asmuth-Bloom method. Interestingly, this was used in [12] as an alternative to SSS for threshold access, as they proved it more efficient.

The present proposal is comparable to the one-time-pad used in the 1960s Cold War era to transmit messages with (at least in principle) information-theoretic security, using random numbers known only to the sender and receiver. That technique does not however allow for expansion into more complex access structures, unlike secret sharing. Effectively, as shown below, such structures allow the secret to be revealed to specified user subsets, whereas other subsets gain no information about the secret message.



(a) (b)

Figure 1. Secret sharing access structures shown as graphs. (a) (3, 3)-SS. (b) Five-share general graph access structure.

Revocation and addition of keys is here analogous to deletion and creation of shares in such a way that security and integrity are maintained. In conventional SSS, new shares are created by simply selecting unused points on the curve and dealing them, but deleting a participant involves removing his/her knowledge of the scheme, hence changing the polynomial. This was solved in [9] by using SSS along with CRT, Vandermonde matrices and "virtual automaton" to enable entering and leaving schemes at will and changing the secret on the fly.

Security in SS has been the subject of much study. Despite being information-theoretically secure, if k is less than n, an adversary having fewer than k shares can indeed narrow the list of potential secrets from the infinite to the finite. In the author's prior work [6], additional alphanumeric keys were suggested to vary the polynomial term used to encode a respective character, and in [8],

different entropic metrics are compared with Kolmogorov complexity [14] in the context of SS to show equivalence between these security metrics and they prove that all entropy metrics are maximal if shares are uniformly randomly generated.

As pointed out in [16], email is inherently non-secure, with the vast majority of email servers leaving messages open to interception and reading of plaintext. Although public-key cryptography is sometimes used, they developed a more efficient protocol, SMEMail. Similarly, an SS solution to messaging security based on computationally secure SS was given in [13], which addresses the problem of creating smaller share sizes with efficient MPC. However, their proposal relies on computationally secure symmetric ciphers. Secret sharing is indeed integral in maintaining perfect security in MPC. In this, the idea is to compute $y = f(x_1, x_2, ...)$, making the result public but concealing the function arguments, shares of which are then distributed.

Shamir's SS has similarly found use in secure multi-cloud, distributing data across servers as random shares. In most cases, such data is first encrypted, then the crypto-keys are shared, but recent work such as [17] has suggested sharing both data and keys. If the data is too large, it is seen as more efficient to merely share keys to the data, but [2] developed a system called CloudStash to share all data, using low-cost cloud storage and multi-threading to improve fault-tolerance. They show that sharing small files is more efficient than sharing keys, and even large files do not incur significant cost.

3 Background to Secret Sharing

Guide to Notation

$p_i \in \mathrm{P}, i = 1, ..., \pi, \pi \in \mathbb{N}$ is the collection of participants.

$s_i \in \mathrm{S}, i = 1, ..., \sigma, \sigma \in \mathbb{N}$ is the collection of characters comprising the message.

$\mathbf{H}^M$ is the collection of master shares.

$\mathrm{H}_i^M \in \mathbf{H}^M$ is the master share held by p_i.

H^T is the transient share conveying a respective message between two participants.

Share elements are denoted using square brackets. For example, $\mathrm{H}^T[i]$ is the value of the *i*th element in the transient share, or $\mathrm{H}^T[j,i]$ is the value of the red channel bitmap pixel in the *i*th column of the jth row.

$\alpha, \beta \in (\mathbb{Z}$ or $\mathbb{R}), \alpha < \beta$ are resp. lower and upper bounds of H^T elements.

$random(X)$ returns a random member of set X, but if it has a trailing superscript, a random matrix of given dimensions is returned.

Φ is a constant known to all $p \in \mathrm{P}$, such that $x_i = \Phi i, i = 1, ..., n$

For simplicity in this section, we consider splitting a secret number s into shares $\mathrm{H}_i, i = 1, ..., n$, where *n* is the number of shares. Any *k* of these must be combined to reveal the secret. This is a threshold (k, n)-SSS scheme, the simplest of which is a (2, 2)-SSS. This sharing requires polynomial,

$$y = \left(\sum_{i=1}^{k-1} r_i x^{k+1-i} \right) + s, r_i = random(\{\alpha, ..., \beta\}) \quad (1)$$

where each $r_i \in \mathbb{Z}$ lies within arbitrarily chosen coefficient bounds α, β.

The dealer randomly selects distinct points $(x_i, y_i), i = 1, ..., n$ on y and hands one to each participant. Combining *k* of these reveals the coefficients, notably s. Shares of s are thus defined as,

$$\mathrm{H}_i = (x_i, y_i), i = 1, ..., n \quad (2)$$

Letting $x_i = \Phi i$ eliminates x-coordinates, decreasing network overhead. In the results presented in this paper, $\Phi = 1$.

The secret can be decoded using Lagrangian interpolation. Coordinates in (2) are used to calculate,

$$y = \sum_{\forall j \in X} \left(y_j \prod_{\forall i \in X} \frac{x - x_i}{x_j - x_i} \right), X \in \Gamma_{qual} \quad (3)$$

where Γ_{qual} is a qualified subset of share indices. The result of (3) is expanded to a polynomial of degree $k-1$. The coefficients of the terms in $x^{k-1}, x^{k-2}, ..., x$ are ignored (as they are random), but the constant is decoded as the secret.

If the secret is a message, each character is converted into its ASCII numeric code, which is encoded separately into its own shares. In this case, the *i*th share of the jth character is denoted $(x_i, y_i)^j$. If $x_i = \Phi i$, it can simply be denoted $(y_i)^j$. That is, $\mathrm{H}_i = (y_i)^1, ..., (y_i)^\sigma$.

4 Proposed Method

In this section, the method for multi-use master share textual secret sharing is proposed. Section 4.1 discusses initialisation, i.e. creation and dealing of master shares, and Section 4.2 discusses the encoding of messages, whereby a transient share is created and transmitted following two obfuscation operations. Finally, Section 4.4 briefly discusses revocation.

4.1 Key Generation

In the present proposal, the keys are analogous to the master shares. The dealer $D \in \mathrm{P}$ designates π as the currently known number of trusted participants and generates a master share for each one. Algorithm 1 summarises key generation.

Algorithm 1: Key Generation through the Creation of Master Shares

Inputs: π

Outputs: $\mathbf{H}^M$

Procedure:

Set $\eta \in \mathbb{N}$ *as the maximum message length*

$\alpha \leftarrow 0, \beta \leftarrow 255$

$w \leftarrow \left\lceil \sqrt{\eta} \right\rceil \in \mathbb{N}$

For $i \leftarrow 1,...,\pi$ *, Do*

$\mathrm{H}_i^M \leftarrow random(\{\alpha,...,\beta\})^{w \times w \times 3} \in \mathbb{N}$

Store H_i^M *as an RGB bitmap image*

End For

$\mathbf{H}^M \leftarrow (\mathrm{H}_1^M,...,\mathrm{H}_\pi^M)$

In the present proposal, shares are stored in raw bitmap files, wherein the value of each pixel in the red channel is the respective share element, as in Figure 2. (Note that lthough this share format is comparable to visual cryptography, bitmaps are used here for the convenience of having different channels for encoding and seeding.)

The green and blue channels are also randomised but it is beneficial to leave these for seeding, which prevents known-plaintext attacks. Furthermore, to prevent related-key attacks, all members of $\mathbf{H}^M$ must be unique and randomly different from all other members. Furthermore, to maintain immunity from interception, $\mathbf{H}^M$ can never be transmitted via the network, but handed over in person.

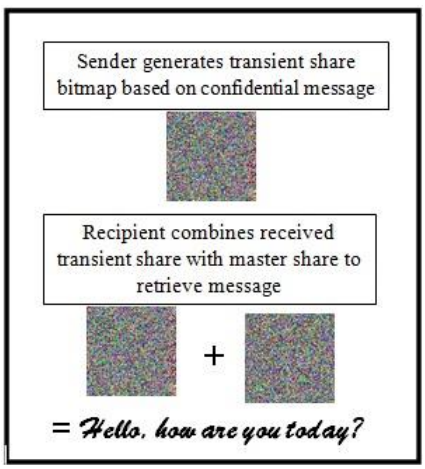


Figure 2. The use of pixel-randomised icons to store and transmit shares as images.

4.2 Message Encoding

4.2.1 Obfuscation Functions

Encoding uses an obfuscated copy of a respective user's master share, allowing it to be securely used multiple times without the possibility of interceptors analysing the repeated use of the same or similar numbers to apply linguistic analysis to them. Algorithm 2 carries out this operation before transmitting a message to the *i*th participant. It uses any cryptographically secure pseudo-random number generator (CSPRNG) initially seeded by the current Unix Timestamp (UT). (*UT* is only input, as opposed to sampled anew, when called later by Algorithm 4 to decode the message.) Shares are contained in the red channel of bitmaps, a row and column index to locate an element.

Algorithm 2: Master Share Obfuscation

Inputs: H_i^M, β, optionally *UT*

Outputs: $\left(\mathrm{H}_i^M\right)'$

Procedure:

Sample UT if not provided as input.

$seed_1 \leftarrow UT$

$seed_2 \leftarrow 0$

$w \leftarrow$ *width of bitmap holding* H_i^M

Set numPixels //5 IS A GOOD VALUE

Seed PRNG with $seed_1$.

Record the date and time components of the UT.

For $j \leftarrow 1,...,numPixels$ **, *do*,**

$j \leftarrow PRNG_{next} \pmod w$

$i \leftarrow PRNG_{next} \pmod w$

$seed_2 \leftarrow seed_2 +$ *24-bit colour value at* $\mathrm{H}_i^M[x, y]$

End For

Seed PRNG with $seed_2$.

For $j \leftarrow 1,...,w$ **, *do*,**

For $k \leftarrow 1,...,w$ **, *do*,**

$\left(\mathrm{H}_i^M\right)'[j,k] \leftarrow \mathrm{H}_i^M[j,k] + 256.PRNG_{next}$

$\left(\mathrm{H}_i^M\right)'[j,k] \leftarrow \left\| \left(\mathrm{H}_i^M\right)'[j,k] \pmod{\beta + 1} \right\|$

End For

End For

There are two stages of pseudo-random number generation. The first relies on *UT*, which is known to an interceptor, who can calculate the pixel positions. However, their values have never been transmitted, hence cannot be known by that individual, and cannot be guessed, as argued in Section 5.2. Furthermore, using a bitmap's red channel leaves other channels never used directly in any prior encoding, so their additional 16 collective bits can be added to the existing 8 bits of the red channel to form a 24-bit number. An arbitrary number of pixel values are collected in this way and summed to produce a secondary seed. The temporarily altered master

share $\left(\mathrm{H}_i^M\right)'$ is generated by a modular sum of random numbers based on this seed.

Given in Section 4.2.2 is the equation for deriving a transient share element from a master share element based on SSS. Repeated use of this equation results in approximately 20% negative correlation between message character ASCII code and transient share element, as well as a mean P-value too small to have confidence in the null hypothesis. This correlation is eliminated by obfuscation of character values and their order, as in Algorithm 3.

Algorithm 3: Message Obfuscation

Inputs: $\mathrm{S}, \mathrm{H}_i^M$, *UT* used in Algorithm 2

Outputs: S' (obfuscated message)

Procedure:

$UT_{seconds} \leftarrow$ *seconds value from UT*

$seed \leftarrow$ *24-bit colour value at* $\mathrm{H}_i^M[1,1] - 100.UT_{seconds}$

Seed PRNG with the calculated value above

Permutate the order of elements in $\left\{s_1', ..., s_{|S|}'\right\}$ *//BASED ON SEEDED PRNG*

Reseed PRNG with the calculated value above

For $j \leftarrow 1, ..., |S|$, ***do,***

$s_j \leftarrow$ *numeric ASCII code of jth character in S*

If $s_j > 94$, ***then*** $s_j' \leftarrow 255 - s_j$, ***End If***

$r_1 \leftarrow random([0, ..., 10])$ *// SEEDED PRNG*

$r_2 \leftarrow random([-7, ..., +7])$ */ SEEDED PRNG*

$s_j' \leftarrow s_j' + r_1$

Rotate bits of s_j' *by* r_2 *positions*

End For

$\mathrm{S}' \leftarrow s_1', ..., s_{|S|}'$

As with share obfuscation, the present operation uses information unknown to an interceptor. Note that the seed is taken from the top-left master share pixel, but the system can use any pixel (preferably one never used for encoding), as long as the position is consistent and known to all participants. The value 94 is selected based on the authors' experimental observation that this is approximately the mean ASCII value of characters comprising passages of text (seemingly varying little in different Roman alphabet languages).

4.2.2 Creation of Transient Shares

Conventional SSS creates the entire share set anew each time a secret is encoded, but in this paper, each user retains a permanent master share, $\mathrm{H}_i^M, i \in \{1, ..., \pi\}$, of which the sender creates an obfuscated version $\left(\mathrm{H}_i^M\right)'$ before transmission.

If $x_i = \Phi i$, each master share element $\left(\mathrm{H}_i^M\right)'[j], j = 1, ..., \eta$ is the coordinate $\left(\Phi, (y_1)^j\right)$ on a Cartesian plane. Coordinate $\mathrm{H}^T[j]$ must be calculated such that an adjoining line crosses the vertical axis at $y = s_j$. Any number of lines can pass through $\mathrm{H}_i^M[j]$ to cross any required point on the *y*-axis, hence allowing reuse of master shares. Figure 3 demonstrates the concept. Here, the intersection point of the two lines is an element a user's master share, and the other two points can be considered elements of transient shares for two different messages.

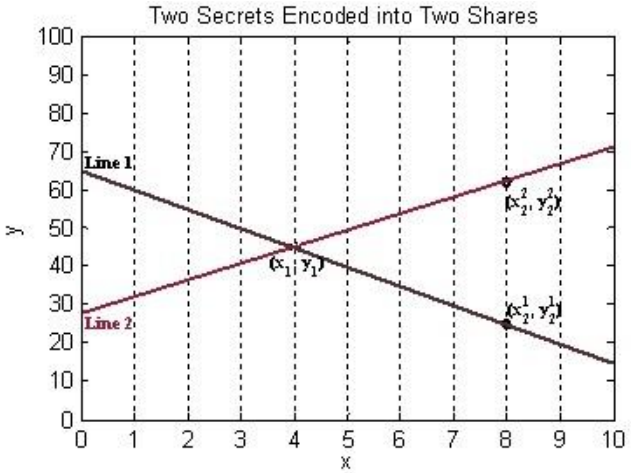


Figure 3. Intersecting lines encoding different ASCII codes simultaneously in two shares.

The gradient of Line 2 for $x_1 \le x \le x_2$ is $\frac{y_2^2 - y_1}{x_2^2 - x_1}$. Since the vertical axis intersect is the constant, the gradient of Line 2 for $0 \le x \le x_1$ is $\frac{y_1 - s_i}{x_1}$. Since these gradients are of the same line, they are equal. Furthermore, (x_1, y_1) and s_i are known, and selecting a value for x_2^2, the value of y_2^2 becomes,

$$y_2^2 = \frac{x_2^2 y_1 + s_i (x_1 - x_2^2)}{x_1}, i = 1, ..., \sigma$$

If $(x_i)^j = \Phi i$ (in Figure 3, $\Phi = 4$), $(x_i)^2 = 2(x_i)^1$, this further reduces (4) to,

$$x_2^j = 2x_1 \Rightarrow y_2^2 = 2y_1 - s_i, i = 1, ..., \sigma$$

Using either of these equations produces an H^T that is maximally entropic. Note in (5) that the ASCII value, effectively a constant, is subtracted from twice y_1, a random number, so y_2^2 must also be random. However, it is clear that larger values of s_i will tend to produce smaller values of y_2, requiring the use of $\left(\mathrm{H}_i^M\right)'$ and S' respectively output by Algorithm 2 and Algorithm 3.

Figure 4 summarises encoding and transmission, wherein p_A sends a confidential message to p_B. Note that the message date and time of creation is also transmitted, along with σ. The latter is necessary for the recipient to determine how many bitmap pixels to decode as ASCII values.

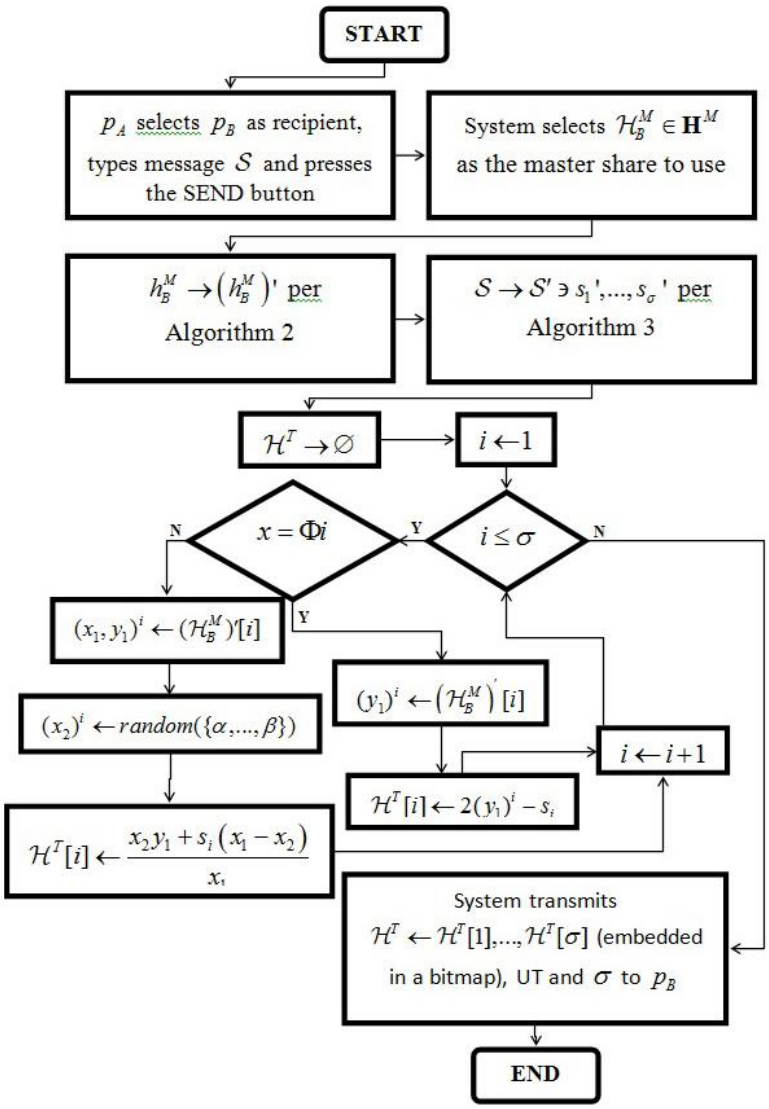


Figure 4. Summary of encoding and transmission

4.3 Message Decoding

Decoding reverses encoding, but alters the recipient master share in the same way as during encoding, hence Algorithm 2 is called. The method is described in Algorithm 4.

Algorithm 4: Message Decoding

Inputs: H^T, received *UT*, σ

Outputs: S

Procedure:

Derive $(\mathrm{H}_i^M)'$ *from* H_i^M *using Alg. 2 with UT as input*

$S \leftarrow \varnothing$

$UT_{seconds} \leftarrow$ *seconds value from received UT*

$seed \leftarrow$ *24-bit value at* $\mathrm{H}_i^M[1,1] - 100.UT_{seconds}$

Seed PRNG with the calculated value above

For $j \leftarrow 1, \ldots, \sigma$, ***do***,

$x_1 \leftarrow \Phi, x_2 \leftarrow 2\Phi$, *where* $x_1, x_2 \in$

$y_1 \leftarrow (\mathrm{H}_i^M)'[j], y_2 \leftarrow (\mathrm{H}^T)'[j]$, *where* $y_1, x_2 \in$

$m \leftarrow \frac{y_2 - y_1}{x_2 - x_1}$

$c \leftarrow y_1 - mx_1$

$r_1 \leftarrow random([0, \ldots, 10])$ *// SEEDED PRNG*

$r_2 \leftarrow random([-7, \ldots, +7])$ *// SEEDED PRNG*

Rotate bits of c $-r_2$ *positions*

$c \leftarrow c - r_1$

If $c > 94$, ***then*** $c \leftarrow 255 - c$, ***End If***

Convert c to ASCII character and append to S

End For

Reseed PRNG with the calculated value above

$\Pi \leftarrow 1,...,\sigma$ *randomly permutated // SEEDED PRNG*

temp $\leftarrow \Pi, S$ *concatenated as two columns*

Sort temp in ascending order by first column

$\mathrm{S} \leftarrow$ *transpose of second column of temp*

Convert ASCII values of S *to characters and display the decoded message*

Algorithm 3 cannot be executed during decoding, as its operations must happen in reverse. Furthermore, to reverse the permutation, the recipient must use the same seed to permutate $1,...,\sigma$ and equate S element indices to those values, reordering the characters to recreate the message sent by p_A.

1.1 Remarks on Revocation

A new trusted recipient requires a master share that is either new or currently not in use. The latter simply requires the dealer to create more shares than participants during initialisation. In this case, all current participants must be made aware of the value of i for $\mathrm{H}_i^M \in \mathbf{H}^M$ corresponding to the newcomer. This can happen in a pre-agreed manner of blanket emailing (using master shares) the requisite details to current participants.

This proposal is intended for a limited number of trusted users (given that $\mathbf{H}^M$ is provided to all users in its entirety). It is possible to generate H_i^M from H_{i-1}^M in a similar way that transient shares are created, by using pixel data in H_i^M to seed PRNGs for the creation of H_{i+1}^M. As this data has never been transmitted and assumed unknown to an interceptor, H_{i+1}^M is by extension unknown. This therefore provides a mechanism for the creation of master shares for newcomers, but still requires a blanket email to inform all current users of the addition.

Removal of a user must be orchestrated by D and involve the cooperation of all remaining users. It is not enough to remove the leaving user's details from the scheme, as that person must still be assumed to have access to $\mathbf{H}^M$ and be able to use it to send messages. It is therefore necessary to permanently alter all shares in $\mathbf{H}^M$ for all participants except the leaver. D organises this by sending seeding data through the system to all remaining users, which they can each use in combination with data from their current master shares, to pseudo-randomly generate and permanently store new shares.

Although these are randomly different from the old ones and an outsider still has no access to any of the data, the process is deterministic. Hence, all participants calculate and store the same share set. (Indeed, this process can be regularly executed to continually alter $\mathbf{H}^M$, so that a message received, for example, the previous day, cannot be decoded the following day using the same master share.)

5 Results and Discussion

5.1 Simulation Results

The parameters used here are $\alpha = 0, \beta = 255, \Phi = 1$, with shares stored as 100×100-pixel bitmaps with a red, green and blue channel. Master shares are randomly initialised, and the first 10 elements of the respective colour channels of H_1^M and H_2^M are as follows:

$$\mathrm{H}_1^M = \begin{pmatrix} (161,174,181,20,112,165,250,54,73,0,...)^{red}, \\ (147,2,29,91,25,116,204,168,9,155,...)^{green}, \\ (78,74,181,80,78,169,4,133,230,108,...)^{blue} \end{pmatrix}$$

$$\mathrm{H}_2^M = \begin{pmatrix} (89,106,254,22,145,144,185,99,125,246,...)^{red}, \\ (33,220,116,91,249,218,88,238,204,22,...)^{green}, \\ (85,222,251,110,104,130,18,229,173,57,...)^{blue} \end{pmatrix}$$

These shares and those of any other participants comprise set $\mathbf{H}^M$, which is physically handed to all participants. At time 10:40:29 on date 14/04/2015, the following short message is typed by p_A to be transmitted to p_B. Given also is the resulting ASCII value sequence.

"*Hello, how are you today?*"

$\rightarrow$ {72, 101, 108, 108, 111, 44, 32, 104, 111, 119, 32, 97, 114, 101, 32, 121, 111, 117, 32, 116, 111, 100, 97, 121, 63}

Therefore, $\sigma = 25$. p_A knows that,

$\mathrm{H}_2^M[1,...,\sigma] = \{89, 106, 254, 22, 145, 144, 185, 99, 125, 246, 209, 205, 124, 154, 94, 247, 216, 63, 119, 217, 31, 214, 226, 3, 55\}^{red}$.

p_A sets the pseudo-random seed to the current date-time (UT), randomly selects five pixels from H_2^M and sums their 24-bit values to produce seed 19558611. He then uses Algorithm 2 to produce altered recipient master share red channel:

$\left(\mathrm{H}_2^M\right)' = \{93,197,214,186,168,71,85,30,135,209,...\}^{red}$

He seeds PRNG in Algorithm 3 based on $UT_{seconds}$ and the 24-bit pixel value at $\mathrm{H}_2^M[1,1]$ and runs the algorithm to create obfuscated message $\mathrm{S}' = \{100, 62, 13, 122, 10, 248, 73, 168, 204, 232,...\}$.

As $\Phi \neq \varnothing$, x-coordinates are fixed, so (5) is used to produce transient share $\mathrm{H}^T = \{86, 332, 415, 250, 326, -106, 97, -108, 66, 186\}$. These values are normalised to lie within $\{0,...,255\}$ and remaining pixel values are padded with random numbers.

This share is transmitted to p_B via email or other messaging system, along with UT and σ. It is crucial to note the recipient has access to the same H_2^M and UT as the sender, therefore seeds Algorithm 3 with the same

value to produce $\left(\mathrm{H}_2^M\right)'$. She then interpolates respective coordinates in her master share and the received transient share, as follows:

Table 1. Calculation of line equations from master and transient shares

Master Share Values	Transient Share Values	Line Equations
89	86	y=-3m+92
106	332	y=226m-120
254	415	y=161m+93
22	250	y=228m-206
145	326	y=181m-36
144	-106	y=-250m+394
185	97	y=-88m+273
99	-108	y=-207m+306
125	66	y=-59m+184
246	186	y=-60m+306

PRNG is re-seeded with the aforementioned value, and the constants of the line equations in Table 1 are randomly altered as in Algorithm 4 to produce the message "H e?lytur wl oaohaode,yo". This was originally permutated based on seed $UT_{seconds}, \mathrm{H}_2^M[1,1]^{24-bits}$, so she randomly permutates sequence $1,\ldots,\sigma$ based on this to produce Π = (1, 2, 17, 15, 24, 4, 21, 13, 6, 9, 14, 23, 11, 3, 8, 12, 22, 18, 25, 5, 20, 7, 19, 16, 10). This and the ASCII values of S' are transposed, column-wise concatenated, and the resulting 25×2 matrix sorted by Π in ascending order. She extracts the S' column, transposes it and converts its values to their character equivalents to produce the original confidential message, S that was sent by p_A.

5.2 Security Analysis

This analysis will begin with definitions of specific attack types, then argue security against those attacks and general attacks, such as cipher-text-only.

5.2.1 Specific Attack Definitions

Definition 1: Exploitation of Common Elements in Transient Shares

Without Algorithm 2 to map $\mathrm{H}_i^M \rightarrow (\mathrm{H}_i^M)'$ when sending a message to the *i*th participant, two messages, s_i^1 and s_i^2, sent at different times, produce identical versions of H^T. Moreover, different messages containing common characters at the same position indices produce identical H^T elements at those positions. Taking account of (5) and Figure 3, this attack is defined by the following procedure:

1) $y_2 = 2y_1 - s_i^1$ and $y_2' = 2y_1 - s_i^2$.
2) $\therefore y_1 = \frac{y_2 - s_i^1}{2}$ and $y_1 = \frac{y_2' - s_i^2}{2}$.
3) $\therefore y_2 - s_i^1 = y_2' - s_i^2$.
4) He rearranges this to derive: $s_i^2 = y_2 - y_2' + s_i^1$.
5) He assumes that s_i^2 contains a string of the one character most likely to be found in the message, for example the letter "e".
6) He calculates the resulting s_i^2 based on the equation in step 4.
7) He eliminates his assumptions about s_i^1 at all position indices where an "unlikely" message character has been derived in s_i^2 (as explained below).
8) He repeats steps 5, 6 and 7 for the second most likely character. He can either leave the already assumed characters as they are, or create new assumptions, some of which might overwrite the previous ones.
9) He continues with assumed characters of descending likelihood, until enough information about the messages is gained to make intelligent guesses as to their contents.

Definition 2: Exploitation of the Correlation Between Share and Message

Conventionally, when two shares are generated, they are individually statistically equivalent to randomly generated sequences, but if one share is generated using (4) or (5) without obfuscation, a significant negative correlation (approximately 20%) results between H^T and *S*.

An interceptor of H^T, knowing this correlation exists, correctly assumes that smaller transient share elements are more likely to encode larger ASCII values and vice versa. He determines the range of values in H^T and decides which values to categorise as small and large. (For instance, small values could be those below half the mean.)

According to [15], "r", "s" and "t" occur with a frequency of 21% in the English language and "e", "a" and "d" occur with a frequency of 25.1%, so he assumes small share values to encode one of the former and large values to encode one of the latter. He might add the space character (ASCII value 32) to the latter, as it occurs with a frequency of 19.2% in English. Although most of these assumptions would be incorrect, he now has a "hook" into the ciphertext that can allow him to carry out further statistical linguistic analysis.

5.2.2 Generic Attacks

Ciphertext Attacks

This attack succeeds if the interceptor accesses sufficient transient shares to decipher the plaintext or derive the correct master share to combine with the present (or future) transient shares to reveal the plaintext. Without

obfuscation, $\mathrm{H}^T[j], j \in 1,...,\sigma$ with the *j*th character identical across the respective messages, are coordinates on a line that pass through the coordinate given in H_i^M for the *i*th recipient. Those transient share elements therefore lie on the same line, which can easily be interpolated to retrieve s_j.

Only two intercepted ciphertexts are required, as respective transient share elements y_2 and y_2' are equal, allowing step 3 of Definition 1 to execute. However, Algorithm 2 randomly alters both these values using seeds calculated from *UT* and H_i^M. The latter has never been seen by the interceptor, so he cannot directly calculate the seed.

Even if the interceptor can predict the correct seed, he cannot execute the second loop in Algorithm 2, as he has no knowledge of H_i^M. He can attempt a brute-force guess at its structure, but because share elements are nothing but polynomial coordinates, y_2^2 can be swapped with y_1 in (5) to produce a version of H_i^M to encode any message up to η characters. Therefore, any brute-force (or otherwise) guess at H_i^M is as good as any other guess.

Since the interceptor has no access to H_i^M, and both shares are needed, by definition, to decode the message, the method is secure against ciphertext attacks.

Known-plaintext Attacks

This attack succeeds if the interceptor obtains a sufficient number of pairs of transient shares and corresponding plaintexts to gain information about future messages by indirect calculation or by constructing knowledge of the *i*th recipient's master share to directly decode those messages. The method presented in this study assumes trusted users who divulge no information about shares or messages to outsiders, but if the interceptor illegally gains access to the requisite messages, he can attempt the attack.

He swaps the vertical coordinates in (5) to calculate $\mathrm{H}_i^M[j] = 2\mathrm{H}^T[j] - s_j, j = 1,...,\sigma$. Without Algorithm 2, this allows him to decode all future messages up to σ characters, but each future message is encoded using $\left(\mathrm{H}_i^M\right)'$, mapped using $seed_2$. However, with the use of the red channel of bitmaps to carry shares and the 24-bit RGB values for seeding, he only has knowledge of a randomly obfuscated version of part of the red channel. He has however no knowledge of the information used to calculate $seed_2$ nor of the original master share on which it is applied.

As discussed in the context of ciphertext attacks, any guess he makes as to the structure of the master share is no better than any other guess. Furthermore, all successive interceptions of shares and their plaintexts likewise only reveal obfuscated data about the red channel with no information about the other channels. Neither the original master share nor future messages can thus be obtained, preventing the attack.

Chosen-plaintext Attacks

The chosen-plaintext attack succeeds if the hacker is able to successfully encode a message of his choosing into a transient share, so that he can gain information about the key. However, the cryptographic key is analogous to H_i^M, to which the outsider does not have access, by the definition of the method. Therefore, he cannot attempt this attack.

Similarly, the chosen ciphertext attack succeeds if the hacker successfully decodes a message from a chosen master share. This is again not possible, by the definition of the method. He can of course derive the correct shares for a plaintext of his choosing, or the correct complementary share and message, given one share, but this is simply the creation of a new SS scheme, bearing no information about the actual master share.

Adaptive Chosen-plaintext and -ciphertext Attacks

For either of these attacks to succeed, the hacker must first carry out a chosen-plaintext (or chosen-ciphertext) attack and adapt future attempts based on the results of previous attempts. However, as already discussed, neither of these attacks is possible, so no adaptation can occur.

Related-key Attacks

The interceptor attempts this attack by obtaining transient shares encoded using $\mathrm{H}_{i_1}^M, \mathrm{H}_{i_2}^M, ...$ encoding the same message and exploiting similarities between the master shares to derive those shares and combine them with intercepted transient shares to decode messages. As discussed in Section 4.1, all members of $\mathbf{H}^M$ must be unique and randomly different from each other, avoiding similarities and rendering the attack moot.

The security against this attack furthermore follows from that of ciphertext attacks, which is equivalent in the present method to attempting to exploit similarities between different obfuscated versions of the same master share.

Frequency Analysis Attacks

This attack succeeds if the interceptor exploits varying frequencies of H^T elements to derive the plaintext based on known linguistic character frequencies. As $\left(\mathrm{H}_i^M\right)'[j]$ is a random number, the resulting $\mathrm{H}^T[j]$ given by (5) is also random, so different instances of the same character are differently encoded with a random relationship

between the encodings, so trivial forms of this type of analysis are not possible.

However, the specific attack given in Definition 2 can be attempted by exploiting the correlation between S and H^T with knowledge of linguistic structure. However, Algorithm 3 not only pseudo-randomly alters the ASCII values of all characters, but shuffles them, destroying any resemblance to a language.

Furthermore, as in Algorithm 2, the PRNG is seeded from UT and H_i^M . As discussed previously, the interceptor can attempt to guess the structure of this share by brute-force or otherwise, but any message up to length η can be encoded using a respective master share, so all possible guesses have equal value.

1.2 Comparative Analysis

1.2.1 Comparisons with Other SS Work

To the best of the authors' knowledge, this is the first study proposing textual secret sharing using SSS as an independent method for confidential messaging, so the following analysis compares this study to other secret sharing research. The most important criterion is master share capability, but also highlighted are access structures, rulebooks (for example basis matrices of visual cryptography), whether a transmitted signal contains an encrypted version of the message (as opposed to a maximally entropic share of it), and the ability to add and/or delete users.

Table 2. Comparative analysis between this study and prior secret sharing work

Study	Master Shares Possible	Threshold Access	General Access	Rulebooks Required	Trans. Contains Encrypted Message	Addition/ Deletion
[20]	No	Yes	No	No	No*	Yes/No
[5]	No	No	No	No	No*	Yes/No
[21]	No	Yes	No	No	No*	Yes/No
[1]	No	Yes	No	No	No*	Yes/No
[18]	No	Yes	No	Yes	No*	No
[11]	Yes	No	No	No	No	No
[22]	Yes	Yes	No	No	No	No
[10]	No	Yes	No	No	No*	Yes/No
[3]	No	Yes	Yes	No	No*	Yes/No
[9]	Yes	Yes	Yes	No	No	Yes
[16]	n/a	n/a	n/a	No	Yes	Yes
[2]	No	Yes	No	Yes	No*	Yes
[6]	No	Yes	No	Yes	Yes	No
[17]	No	Yes	No	No	No*	Yes
present proposal	Yes	Yes	No	No	No	Yes

*Note that although the secret sharing methods in Table 2 necessarily produce maximally entropic shares impossible to individually decipher, those that lack master share capability necessitate the transmission of all shares, likely at different times to the respective users. In this case, enough intercepted shares of the same secret leak that secret. Equivalently, the Cloud applications in [2] and [17] are vulnerable if the various Cloud servers are simultaneously infiltrated.

The proposed method does not currently support access structure. Although they not need for a straight-forward communication between two individuals, they would be useful for more complex communications, for example if the recipient requires both her own and her manager's master share to decode the message.

6 Conclusion and Further Work

This paper has proposed Shamir's Secret Sharing, an information-theoretically secure cipher, as an independent method to secure textual messages such that the transmitted signal only contains a random share of the message. Each participant is handed a copy (itself optionally encrypted) of the master share set. The sender uses the data in the recipient's master share and message data to generate a transient share for transmission. The recipient combines this with her master share to reveal the message. Master shares are fully reusable, because each transient share results from two levels of random obfuscation, with the PRNG seeded from data in the recipient's master share. By definition, an outsider has not seen this, but might attempt to guess it by brute force or otherwise. However, he can derive a share that correctly

decodes any sequence of characters from the intercepted transient share, so all guesses have equal value.

An advantage of secret sharing is access structures, as shown in [3] for sharing data and [4] for sharing images. This study is limited to (2, 2) schemes, but ongoing work focuses master and transient shares for access structures, such that third parties can be involved in decoding. Further work will also address trust among users. This will use both conventional ciphers to encrypt $\mathbf{H}^M$, as well as use SS to share $\mathbf{H}^M$ data between participants, such that the system combines shares of shares across the network before those shares, in turn, combine with the transient share to decode the message.

References

[1] Aldosary, S., Howells, G. A robust multimodal biometric security system using the polynomial curve technique within Shamir's Secret Sharing algorithm, Proceedings of Third International Conference on Emerging Security Technologies, 2012; 66-69

[2] Alsolami, F., Boult, T. CloudStash: Using Secret-Sharing Scheme to Secure Data, Not Keys, in Multi-Clouds, Proceedings of 11th International Conference on Information Technology: New Generations, 2014; 315-320

[3] Asmuth, M., Bloom, J. A modular approach to key safeguarding. IEEE Transactions on Information Theory 1983, 29, 566-584

[4] Ateniese, G., Blundo, C., De Santis,A. Constructions and Bounds for Visual Cryptography. Automata, Languages and Programming 1996, volume 1099, 416-428

[5] Blakley, G.R. Safeguarding cryptographic keys, Proceedings of the National Computer Conference, 1979; 313-317

[6] Buckley, N., Nagar, A.K., Arumugam, S. Key-based Steganographic Textual Secret Sharing, Proceedings of Third International Conference on Soft Computing for Problem Solving, 2014; 25-34

[7] Daemen, J., D-Wave Systems, 2014, http://www.dwavesys.com/ d-wave-two-system

[8] Dai, S., Guo, D. Comparing Security Notions of Secret Sharing Schemes. MDPI Entropy 2015, 17, 1135-1145

[9] Dolev, S., Lahiani, L., Yung, M. Secret swarm unit: Reactive k-secret sharing. Ad Hoc Networks 2012, 10, 1291-1305

[10] Ge, L., Tang, S. Sharing Multi-secret Based on Circle Properties, Proceedings of 2008 International Conference on Computational Intelligence and Security, 2008; 340-344

[11] Kafri, O., Keren, E. Image encryption by multiple random grids. Optical Letters 1987, 12, 377-379

[12] Kaya, K., Selcuk, A.A. Threshold Cryptography Based on Asmuth-Bloom Secret Sharing.2007, 1-20

[13] Kikuchi, R., Chida, K., Ikarashi, D., Hamada, K., Takahashi, K. Secret Sharing Schemes with Conversion Protocol to Achieve Short Share-Size and Extendibility to Multiparty Computation. In *Visual Cryptography and Secret Image Sharing*; Boyd, C. & Simpson, L., Eds; Publisher: Springer-Verlag, 2013; pp. 419-434

[14] Kolmogorov, A.N. Three approaches to the quantitative definition of information. International Journal of Computer Mathematics 1968, 2, 157-168

[15] Micka, P., 2008, http://en.algoritmy.net/ article/ 40379/Letter-frequency-English

[16] Mohsen, T. SMEmail – A New Protocol for the Secure E-mail in Mobile Environments, Proceedings of the Australian Telecommunications Networks and Applications Conference (ATNAC'08), 2008; 39-44

[17] Muhil, M., Krishna, U.H., Kumar, R.K., Mary Anita, E.A. Securing Multi-Cloud using Secret Sharing Algorithm, Proceedings of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15), 2015; 421-426

[18] Naor, M., Shamir, A. Visual Cryptography, Proceedings of EUROCRYPT 1994, 1994; 1-12

[19] Rivest, R.L., Shamir, A., Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 1978, 21, 120-126

[20] Shamir, A. How to Share a Secret. Communications of the ACM 1979, 22, 612-613

[21] Ulutas, M., Ulutas, G., Nabiyev, V.V. Medical image security and EPR hiding using Shamir's secret sharing scheme. The Journal of Systems and Software 2011, 84, 341-353

[22] Wu, X., Sun, W. Improving the visual quality of random grid-based visual secret sharing. Signal Processing 2013, 93, 977-995