SECRET SHARING ALGORITHM'S

SHAMIR 'S SECRET SHARING ALGORITHM

COMPLEXITY

The only complexity of Shamir's approach in a cloud computing model is that the amount of storage required is increased by n times.

EFFICIENCY

(2) When k is kept fixed, Di pieces can be dynamically added or deleted (e.g., when executives join or leave the company) without affecting the other D, pieces. (A piece is deleted only when a leaving executive makes it completely inaccessible, even to himself.)

(3) It is easy to change the Di pieces without changing the original data D -- all we need is a new polynomial q(x) with the same free term. A frequent change of this type can greatly enhance security since the pieces exposed by security breaches cannot be accumulated unless all of them are values of the same edition of the q(x) polynomial.

(4) By using tuples of polynomial values as Di pieces, we can get a hierarchical scheme in which the number of pieces needed to determine D depends on their importance. For example, if we give the company's president three values of q(x), each vice-president two values of q(x), and each executive one value of q(x), then a (3, n) threshold scheme enables checks to be signed either by any three executives, or by any two executives one of whom is a vice-president, or by the president alone.

SECURITY

Shamir Secret Sharing

Part of the challenge in designing a secret-sharing scheme is making sure that *any* authorized set of users can reconstruct the secret. We have just seen that *any* $d + 1$ points on a degree-d polynomial are enough to reconstruct the polynomial. So a natural approach for secret sharing is to let each user's share be a point on a polynomial.

That's exactly what **Shamir secret sharing** does. To share a secret $m \in \mathbb{Z}_p$ with threshold t, we choose a degree-$(t-1)$ polynomial f that satisfies $f(0) \equiv_p m$, with all other coefficients chosen uniformly in $\mathbb{Z}_p$. The ith user receives the point $(i, f(i)\%p)$ on the polynomial. The interpolation theorem shows that any t shares can uniquely determine the polynomial f, and hence recover the secret $f(0) \equiv_p m$.

Theorem 3.9 *Shamir's secret-sharing scheme (Construction 3.7) is secure according to Definition 3.2.*

Proof Let $\mathcal{S}$ denote the Shamir secret-sharing scheme. We prove that $\mathcal{L}^{\mathcal{S}}_{\text{tsss-L}} \equiv \mathcal{L}^{\mathcal{S}}_{\text{tsss-R}}$ via a hybrid argument.

$\mathcal{L}^{\mathcal{S}}_{\text{tsss-L}}$:

$\mathcal{L}^{\mathcal{S}}_{\text{tsss-L}}$
$\text{QUERY}(m_L, m_R, U)$:
if $\lvert U\rvert \geqslant t$: return err
$f_1, \ldots, f_{t-1} \leftarrow \mathbb{Z}_p$
$f(x) := m_L + \sum_{j=1}^{t-1} f_j x^j$
for $i \in U$:
$s_i := (i, f(i) \,\%\, p)$
return $(s_i)_{i \in U}$

Our starting point is $\mathcal{L}^{\mathcal{S}}_{\text{tsss-L}}$, shown here with the details of Shamir secret-sharing filled in.

$\text{QUERY}(m_L, m_R, U)$:
return $\text{QUERY}'(m_L, t, U)$

$\diamond$

$\mathcal{L}_{\text{ssss-real}}$
$\text{QUERY}'(m, t, U)$:
if $\lvert U\rvert \geqslant t$: return err
$f_1, \ldots, f_{t-1} \leftarrow \mathbb{Z}_p$
$f(x) := m + \sum_{j=1}^{t-1} f_j x^j$
for $i \in U$:
$s_i := (i, f(i) \,\%\, p)$
return $(s_i)_{i \in U}$

Almost the entire body of the QUERY subroutine has been factored out in terms of the $\mathcal{L}_{\text{ssss-real}}$ library defined above. The only thing remaining is the "choice" of whether to share m_L or m_R. Restructuring the code in this way has no effect on the library's behavior.

return $(s_i)_{i \in U}$

return $(s_i)_{i \in U}$

In other words, if we evaluate a uniformly chosen degree $t-1$ polynomial on fewer than t points, the results are (jointly) uniformly distributed.

Proof We will prove the lemma here for the special case where the calling program always provides a set U with $\lvert U\rvert = t-1$. Exercise 3.5 deals with the more general case.

Fix a message $m \in \mathbb{Z}_p$, fix set U of users with $\lvert U\rvert = t-1$, and for each $i \in U$ fix a value $y_i \in \mathbb{Z}_p$. We wish to consider the probability that a call to $\text{QUERY}(m, t, U)$ outputs $\big((i, y_i)\big)_{i \in U}$, in each of the two libraries.

In library $\mathcal{L}_{\text{ssss-rand}}$, this event happens with probability $1/p^{t-1}$ since QUERY chooses the $t-1$ different y_i values uniformly in $\mathbb{Z}_p$.

In library $\mathcal{L}_{\text{ssss-real}}$, the event happens if and only if the degree-$(t-1)$ polynomial $f(x)$ chosen by QUERY happens to pass through the set of points $\mathcal{P} = \{(i, y_i) \mid i \in U\} \cup \{(0, m)\}$. These are t points with distinct x-coordinates, so by Theorem 3.6 there is a *unique* degree-$(t-1)$ polynomial f with coefficients in $\mathbb{Z}_p$ passing through these points.

The QUERY subroutine picks f uniformly from the set of degree-$(t-1)$ polynomials satisfying $f(0) \equiv_p m$, of which there are p^{t-1}. Exactly one such polynomial causes the event in question, so the probability of the event is $1/p^{t-1}$.

Since the two libraries assign the same probability to all outcomes, we have $\mathcal{L}_{\text{ssss-real}} \equiv \mathcal{L}_{\text{ssss-rand}}$. ∎

$\text{QUERY}(m_L, m_R, U)$:
  return $\text{QUERY}'(m_L, t, U)$

$\diamond$

$\mathcal{L}_{\text{ssss-rand}}$

$\text{QUERY}'(m, t, U)$:
  if $|U| \geqslant t$: return err
  for $i \in U$:
    $y_i \leftarrow \mathbb{Z}_p$
    $s_i := (i, y_i)$
  return $(s_i)_{i \in U}$

By Lemma 3.8, we can replace $\mathcal{L}_{\text{ssss-real}}$ with $\mathcal{L}_{\text{ssss-rand}}$, having no effect on the library's behavior.

$\text{QUERY}(m_L, m_R, U)$:
  return $\text{QUERY}'(\boxed{m_R}, t, U)$

$\diamond$

$\mathcal{L}_{\text{ssss-rand}}$

$\text{QUERY}'(m, t, U)$:
  if $|U| \geqslant t$: return err
  for $i \in U$:
    $y_i \leftarrow \mathbb{Z}_p$
    $s_i := (i, y_i)$
  return $(s_i)_{i \in U}$

The argument to QUERY' has been changed from m_L to m_R. This has no effect on the library's behavior, since QUERY' is actually ignoring its argument in these hybrids.

$\text{QUERY}(m_L, m_R, U)$:
  return $\text{QUERY}'(m_R, t, U)$

$\diamond$

$\mathcal{L}_{\text{ssss-real}}$

$\text{QUERY}'(m, t, U)$:
  if $|U| \geqslant t$: return err
  $f_1, \ldots, f_{t-1} \leftarrow \mathbb{Z}_p$
  $f(x) := m + \sum_{j=1}^{t-1} f_j x^j$
  for $i \in U$:
    $s_i := (i, f(i) \,\%\, p)$
  return $(s_i)_{i \in U}$

Applying the same steps in reverse, we can replace $\mathcal{L}_{\text{ssss-rand}}$ with $\mathcal{L}_{\text{ssss-real}}$, having no effect on the library's behavior.

$\mathcal{L}^{\mathcal{S}}_{\text{tsss-R}}$:

$\mathcal{L}^{\mathcal{S}}_{\text{tsss-R}}$

$\text{QUERY}(m_L, m_R, U)$:
  if $|U| \geqslant t$: return err
  $f_1, \ldots, f_{t-1} \leftarrow \mathbb{Z}_p$
  $f(x) := m_R + \sum_{j=1}^{t-1} f_j x^j$
  for $i \in U$:
    $s_i := (i, f(i) \,\%\, p)$
  return $(s_i)_{i \in U}$

A subroutine has been inlined, which has no effect on the library's behavior. The resulting library is $\mathcal{L}^{\mathcal{S}}_{\text{tsss-R}}$.

We showed that $\mathcal{L}^{\mathcal{S}}_{\text{tsss-L}} \equiv \mathcal{L}_{\text{hyb-1}} \equiv \cdots \equiv \mathcal{L}_{\text{hyb-4}} \equiv \mathcal{L}^{\mathcal{S}}_{\text{tsss-R}}$, so Shamir's secret sharing scheme is secure. ■

DRAWBACKS/POINT OF FAILURE

The security flaw in this method is that, if the data exhibits some pattern frequently, and that the attacker gets hold of m < k slices, there are great possibilities for him gaining the secret S.
For Shamir's scheme, storage and bandwidth requirements are multiplied by N
– E.g., 5 shares for 1 TB of data requires 5 TB raw
– For Blakely's method, it is multiplied by (N · K)
• Encoding time per byte grows with N · K
– Encoding for 3-of-5 is 10X faster than a 10-of-15
• These forms of secret sharing are unsuitable for performance- or cost-sensitive bulk data storage.

OUTPUT

Example 1

Example. Shamir secret sharing with $p = 31$. Let the threshold be $t = 3$, and the secret be $7 \in \mathbb{Z}/31\mathbb{Z}$. We choose elements at random $a_1 = 19$ and $a_2 = 21$ in $\mathbb{Z}/31\mathbb{Z}$, and set $f(x) = 7 + 19x + 21x^2$. As the trusted pary, we can now generate as many shares as we like,

$$\begin{array}{ll} (1, f(1)) = (1, 16) & (5, f(5)) = (5, 7) \\ (2, f(2)) = (2, 5) & (6, f(6)) = (6, 9) \\ (3, f(3)) = (3, 5) & (7, f(7)) = (7, 22) \\ (4, f(4)) = (4, 16) & (8, f(8)) = (8, 15) \end{array}$$

which are distributed to the holders of the share recipients, and the original polynomial $f(x)$ is destroyed. The secret can be recovered from the formula

$$f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \le i \le t \\ i \ne j}} \frac{x - x_j}{x_i - x_j} \quad \Rightarrow \quad f(0) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \le i \le t \\ i \ne j}} \frac{x_j}{x_j - x_i}$$

using any t shares $(x_1, y_1), \ldots, (x_t, y_t)$. If we take the first three shares $(1, 16)$, $(2, 5)$, $(3, 5)$, we compute

$$\begin{aligned} f(0) &= \frac{16 \cdot 2 \cdot 3}{(1-2)(1-3)} + \frac{5 \cdot 1 \cdot 3}{(2-1)(2-3)} + \frac{5 \cdot 1 \cdot 2}{(3-1)(3-2)} \\ &= 3 \cdot 2^{-1} + 15 \cdot (-1) + 10 \cdot 2^{-1} = 17 - 15 + 5 = 7. \end{aligned}$$

This agrees with the same calculation for the shares $(1, 16)$, $(5, 7)$, and $(7, 22)$,

$$\begin{aligned} f(0) &= \frac{16 \cdot 5 \cdot 7}{(1-5)(1-7)} + \frac{7 \cdot 1 \cdot 7}{(5-1)(5-7)} + \frac{22 \cdot 1 \cdot 5}{(7-1)(7-5)} \\ &= 2 \cdot 24^{-1} + 18 \cdot (-8)^{-1} + 17 \cdot 12^{-1} = 13 + 21 + 4 = 7. \end{aligned}$$

Example 2

Share Computation

Step1: Decide secret

First, we need to decide our secret message. To make our story simple, let's choose dead simple one, 3, as our secret.

If you choose more complex message such as *I love you* or *4b0649b1faf1c1ea7cb0e900*, you just need to convert them to byte array so that you can treat them as a number.

Step2: Decide threshold

Next thing to decide is the threshold. We will choose 3 as our threshold. This means that you need at least three shares to recover the secret.

Step3: Create polynomial

We need to create our polynomial. Polynomial is the equation that looks like $y=3x+1$ or $y=5x^2+10x-3$. You can choose any numbers for coefficient, but the degree of your polynomial must be threshold -1 .

Our threshold is 3, so the degree must be 2 in our case. The polynomial of degree of 2 should takes the form of $y=ax^2+bx+c$. Since you can choose any numbers for coefficient, we will use 2 for a and 1 for b.

What c will be? c must to be our secret. Therefore, we use 3 for c.

This is our polynomial: $y=2x^2+x+3$.

Now we have everything to demonstrate Shamir's secret sharing. This is our configuration

Secret: 3

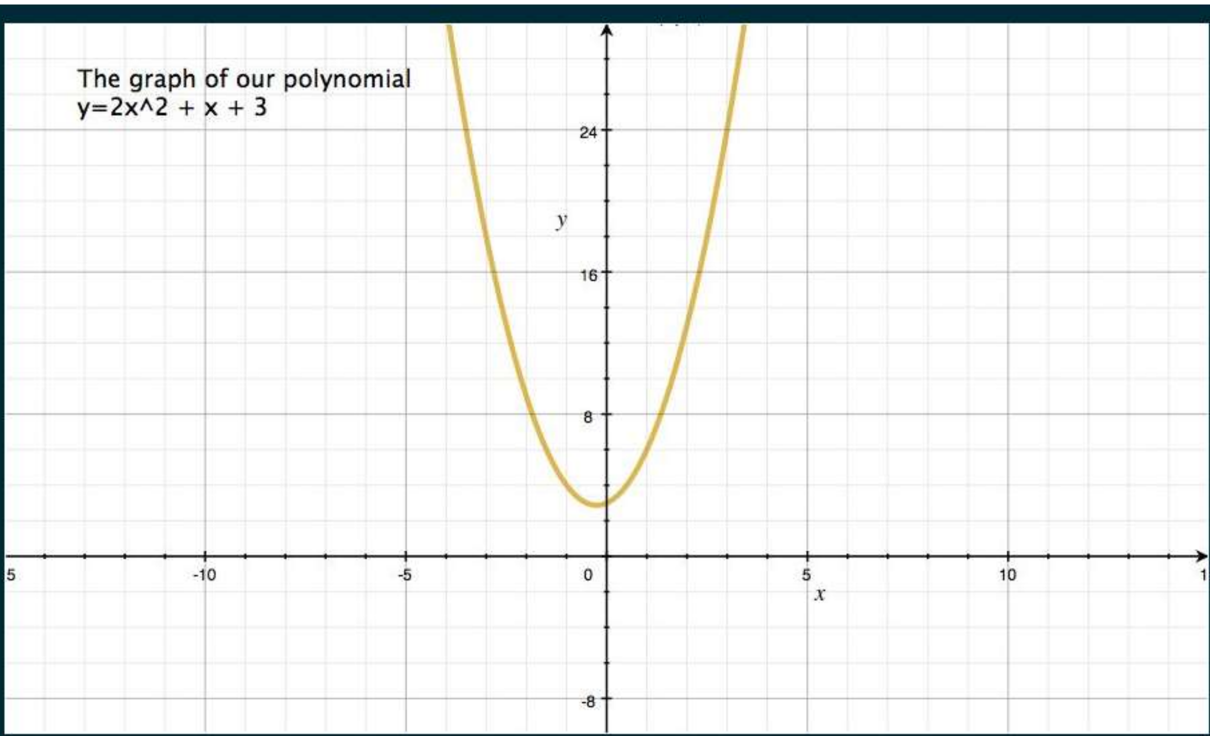
Threshold: 3

Polynomial: $y=2x^2+x+3$

Step4: Draw graph

Note that drawing graph is not necessary to do computation for Shamir's secret sharing. However, we can understand how this works better by drawing graph.

The graph of our polynomial looks like this one.



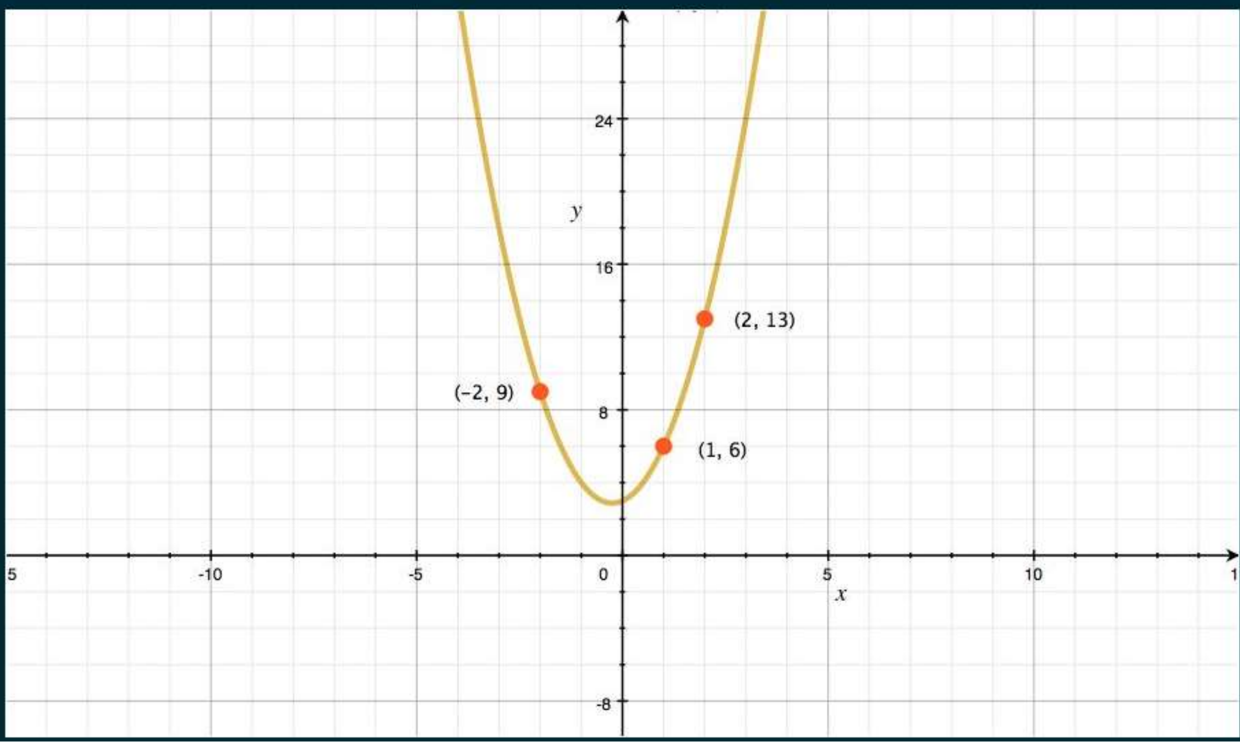
Step5: Plot points on the graph

Plot three points on the line of graph.

Step5: Plot points on the graph

Plot three points on the line of graph.

For example, we plot (x,y)(1, 6), (x,y)(2,13), (x,y)(-2, 9) on the line of graph.



These points are your *shares*. The value in x is called *share number* and the value of y is a *share*.

Remember that we decided our threshold to be 3? Three shares are minimum number of shares that we need. That's why we plotted three points.

You can get even more shares by plotting extra points if you want to distribute shares to more parties.

Once you get shares, forget everything but your threshold and shares!! Throw your polynomial, the graph, and your secret.

As long as you have your shares and threshold, you can recover everything else.

Secret Reconstruction

Now you know nothing but shares and threshold, but you can still recover your secret by combining the shares.

To get this idea, we will again use graph.

Plot points of your shares, (x,y)(1, 6), (x,y)(2,13), (x,y)(-2, 9), on the graph. Then connecting your points and draw imaginary line. If you could do this, you can now recover your polynomial because you are connecting points derived from the polynomial.

Not sure if your line is accurate? Yes, it is is difficult to draw the completely same line because it is curving line. It doesn't matter because drawing graph is just to help you understand the idea.

However, by definition, 2 points are sufficient to define a line, 3 points are sufficient to define a parabola (Wikipedia)

Therefore, although it is difficult to draw the imaginary line by hands from points of shares, you should be able to do that by doing some math. Now, let's do this.

Polynomial interpolation

Since we know that our threshold is 3, we know that we need to get the polynomial of degree of 2 (remember that degree is threshold - 1) which looks like this: $y=ax^2+bx+c$

Substitute three points into $y=ax^2+bx+c$.

(1) (1,6) => c = a + b - 6

(2) (2,13) => c = 4a + 2b - 13

(3) (-2,9) => c = 4a - 2b - 9

Now substitute (1) into (2) and (3) to get a.

(4) substitute (1) into (2) => b = -3a + 7

(5) substitute (1) into (3) => 3b = 3a -3

(6) substitute (4) into (5) => a = 2

We could get a. Let's compute b next.

(7) substitute a=2 into (1) => c=b - 4

```
    (8) substitute a=2 into (2) => c=2b - 5

    (9) substitute (7) into (8) => b = 1

Now we could get a and b. At last, we can compute c.

    (10) substitute a=2 and b=1 into (1) => c=3

We are done. We could recover original polynomial y=2x^2+x+3 and you can find your
secret at free coefficient, which is 3.
```

RABBIN'S INFORMATION DISPERSAL ALGORITHM

EFFICIENCY

3. The size of pieces

The efficiency of any information dispersal algorithm, is computed regarding the size of pieces given to each participant. So, even in the case of general access structures, we are interested in minimizing the size of fragments distributed to participants.

3.1. Information reduction

If we are interested in limiting the maximum size of fragments for each participant (i.e., the maximum quantity of information that must be given to any participant), then a worst-case measure of the maximum of $H(G_i)$ over all $P_i \in \mathscr{P}$ naturally arises.

Definition 3.1. We define the *information reduction* of an information dispersal algorithm Σ for the access structure $\mathscr{A}$, when the probability distribution on the set of files F is Π_F, as

$$\varrho(\mathscr{A}, \Pi_F, \Sigma) = \frac{H(F)}{\max\{H(G_i):\ 1 \leqslant i \leqslant n\}}.$$

The following theorem proves an upper bound on the information reduction above defined.

Theorem 3.1. *Let $\mathscr{A}$ be an access structure on a set $\mathscr{P}$ of n participants. The information reduction of any information dispersal algorithm Σ for $\mathscr{A}$ satisfies*

$$\varrho(\mathscr{A}, \Pi_F, \Sigma) \leqslant \varrho_{\max},$$

where $\varrho_{\max} = \min\{|A|: A \in \mathscr{A}^0\}$.

Proof. Let $A \in \mathscr{A}^0$ such that $|A| = \varrho_{\max}$. From Lemma 2.1, any information dispersal algorithm Σ for F must give to at least a participant $P_j \in A$ a fragment such that $H(G_j) \geqslant H(F)/\varrho_{\max}$. So, for any information dispersal algorithm Σ, $\max\{H(G_i): 1 \leqslant i \leqslant n\} \geqslant H(G_j) \geqslant H(F)/\varrho_{\max}$. Hence, we obtain

$$\varrho(\mathscr{A}, \Pi_F, \Sigma) = \frac{H(F)}{\max\{H(G_i): 1 \leqslant i \leqslant n\}} \leqslant \varrho_{\max},$$

which proves the theorem. $\square$

3.2. *Average information reduction*

In many cases it is preferable to limit the sum of the size of fragments given to all participants. In such a cases the arithmetic mean of the size of fragments for each participant is a more appropriate measure.

The following theorem was proved by Naor and Roth [12] in their special case, and shows a relation between the size of fragments of participants and the solution $M_{\min}^{(1)}$ of the linear problem. It can be applied without any changes to our scenario. When adapted to our notation their theorem is the following.

Theorem 3.2 (Naor and Roth [12]). *For any access structure $\mathscr{A}$ on a set $\mathscr{P}$ of n participants, and for any file in F having entropy $H(F)$, the size $H(G_i)$ of fragments distributed to participants by any information dispersal algorithm Σ satisfies*

$$M_{\min}^{(1)} \cdot H(F) \leqslant I(\mathscr{A}, \Pi_F) \leqslant \sum_{i=1}^{n} H(G_i),$$

where $I(\mathscr{A}, \Pi_F)$ is the integer solution to the optimization problem $IP1$:

$$\boxed{\begin{array}{lll} \textit{Minimize} & M = \sum_{i=1}^{n} \alpha_i & \\ \textit{subject to} & \alpha_i \geqslant 0,\ \alpha_i\ \textit{integer}, & 1 \leqslant i \leqslant n \\ & \sum_{P_i \in A} \alpha_i \geqslant H(F), & \forall A \in \mathscr{A}^0 \end{array}}$$

From Theorem 3.2 the next corollary easily follows which proves an upper bound on the average information reduction above defined using the optimal solution $M_{\min}^{(1)}$ of the linear problem $LP1$.

The following theorem was proved by Naor and Roth [12] in their special case, and shows a relation between the size of fragments of participants and the solution $M_{\min}^{(1)}$ of the linear problem. It can be applied without any changes to our scenario. When adapted to our notation their theorem is the following.

Theorem 3.2 (Naor and Roth [12]). *For any access structure $\mathscr{A}$ on a set $\mathscr{P}$ of n participants, and for any file in F having entropy $H(F)$, the size $H(G_i)$ of fragments distributed to participants by any information dispersal algorithm Σ satisfies*

$$M_{\min}^{(1)} \cdot H(F) \leqslant I(\mathscr{A}, \Pi_F) \leqslant \sum_{i=1}^{n} H(G_i),$$

where $I(\mathscr{A}, \Pi_F)$ is the integer solution to the optimization problem $IP1$:

$$\boxed{\begin{array}{lll} \textit{Minimize} & M = \sum_{i=1}^{n} \alpha_i & \\ \textit{subject to} & \alpha_i \geqslant 0,\ \alpha_i\ \textit{integer}, & 1 \leqslant i \leqslant n \\ & \sum_{P_i \in A} \alpha_i \geqslant H(F), & \forall A \in \mathscr{A}^0 \end{array}}$$

From Theorem 3.2 the next corollary easily follows which proves an upper bound on the average information reduction above defined using the optimal solution $M_{\min}^{(1)}$ of the linear problem $LP1$.

COMPLEXITY

Information dispersal algorithm (IDA) was first introduced by Rabin [1], [2] in 1989. The (n, k) IDA transforms a digital source file into n smaller files (shadows), and the receipt of any k out of the n shadows can reconstruct the source file. By such coding technology, the robustness and fault-tolerance of important files can be improved in communication and storage systems. The idea of IDA is similar to the polynomial secret sharing [35], [36], [37]. However, IDA does not consider the security issue in the design of coding system. In a well-designed IDA, the length of each shadow is (asymptotically) equal to one k-th of the length of source file. Precisely, the length of each shadow achieves the theoretical lower bound [3], under the condition of maximal entropy in the source file. The IDA had been applied to many applications, e.g., distributed data storage [4], RAID codes [5], peer-to-peer techniques [6], multicast [7], and secret sharing [33]. A remarkable coding technique, namely the fountain code [34], is the alternative technique to disperse the source file in the network environment.

Conceptually, the (n, k) IDA can be treated as the (n, k) erasure code. When the receiver acquires k shadows, the scenario is equivalent to erasing the corresponding $n - k$ non-received symbols in an n-symbol codeword. Thus, the optimal erasure codes, such as Maximum Distance Separable (MDS) codes [15], [17], [18] or Reed-Solomon(RS) codes [19], [20],

[22], can serve as the applicable coding systems for the (n, k) IDA. The standard implementation of (n, k) RS erasure codes require $O(nk)$ operations in encoding and $O(k^2)$ operations in decoding via ordinary matrix multiplications.

The computational complexity is a challenge in IDA. The fast IDA can improve the throughput of the real-time systems by demanding large amount of encoding and decoding operations. Various IDAs have been reported in previous literature. In many cases, the erasure codes (or IDAs) can be formulated as the matrix-product forms, so the encoding and decoding complexities depend on the overhead of computing the matrix products. The conventional approaches of (n, k) IDA, such as [1], do not utilize the fast techniques on encoding and decoding processes. Therefore, the conventional encoding algorithm requires $O(nk)$ operations and the conventional decoding algorithm requires $O(k^2)$ operations. To further reduce the computational overhead, the fast Fourier transforms (FFT) over finite field with characteristic two [8], [9], [10], [11] or the fast Fermat number transforms (FNT) are employed in the coding algorithms. For example, Preparata [12] presented the realization of the coding schemes using FFT over finite fields, and the computational complexities are $O(n \log n)$ in encoding and $O(k(n - k + \log k))$ in decoding. Dianat and Marvasti [13] presented the systematic and nonsystematic codes of puncturing RS codes based on FFT over finite fields. Soro and Lacan [14] proposed a Reed-Solomon erasure coding algorithm with complexity $O(n \log n)$ in both encoding and decoding. Lacan and Fimes [15] investigated a systematic MDS erasure coding algorithm based on Vandermonde matrices. For the case $k/n \geq 1/2$, Lin and Chung [38] present a (n, k) IDA with complexities $O(n \log(n-k))$ in encoding and decoding. For the finite field $GF(2^r)$ with characteristic two, Didier [16] presented an decoding algorithm for RS erasure codes $O(2^r \log^2 2^r)$ via fast Walsh transforms. In Section VII, we compare the proposed IDA with those existing methods.

There exist works for erasure codes. G. L. Feng et al. [17], [18] proposed (n, k) MDS codes based on exclusive-OR (XOR) operations. Truong et al. [19] proposed a fast RS decoding algorithm for correcting both errors and erasures. By FFT over finite field with characteristic two [10], [11], the FFT version of RS decoding algorithm had been proposed [20]. Lin et al. [21] proposed a fast algorithm for computing the syndromes of RS codes. Note that there exist several non-optimal erasure codes, such as [34] and [23], [24]. In general, those non-optimal erasure codes require lower computational complexities than optimal erasure codes. However, non-optimal erasure codes cannot guarantee successful decoding from arbitrary k out of the n codeword symbols, as opposed to the guaranteed successful decoding in the optimal erasure codes.

By our survey on the (n, k) erasure coding algorithms over Fermat field, the best records of encoding algorithm take $O(n \log n)$ operations [12], [13], [14] and the decoding algorithms take $O(n \log n)$ [14] or $O(k \log^2 k)$ [15] operations. In this paper, we propose a fast (n, k) IDA based on erasure Reed-Solomon coding systems over Fermat fields. Given k source symbols, the proposed encoding algorithm requires complexity $O(n \log k)$, which is lower than the existing work $O(n \log n)$. In decoding, we present two algorithms. Given k shadow symbols, the main procedure of first decoding algorithm requires complexity $O(n \log k)$, which is lower than the existing work $O(n \log n)$ [12], [13], [14]. The main procedure of second decoding algorithm requires complexity $O(k \log^2 k)$ with smaller leading constant than the fast polynomial interpolation algorithm [15], [25]. By the computational complexities of the two proposed decoding algorithms, the first decoding algorithm is applicable for $n \leq k \log^2 k$, and otherwise the second decoding algorithm is to be adopted. The criteria of choosing one of the two decoding algorithms are discussed in Section VIII-B. It is noted that the [38] presents the IDA for the high code rates $k/n \geq 1/2$. In contrast, the proposed IDA is suitable for the low code rates $k/n \leq 1/2$, and the detailed comparisons between [38] and the proposed method are detailed in Section VII. The potential applications of the low-rate codes are in the communication systems. For example, the [39] indicates that the low-rate codes are applicable to code-division multiple-access (CDMA) systems.

SECURITY

Proposed by Michael O. Rabin in 1989 as a method to achieve efficiency, security, load balancing and fault tolerance

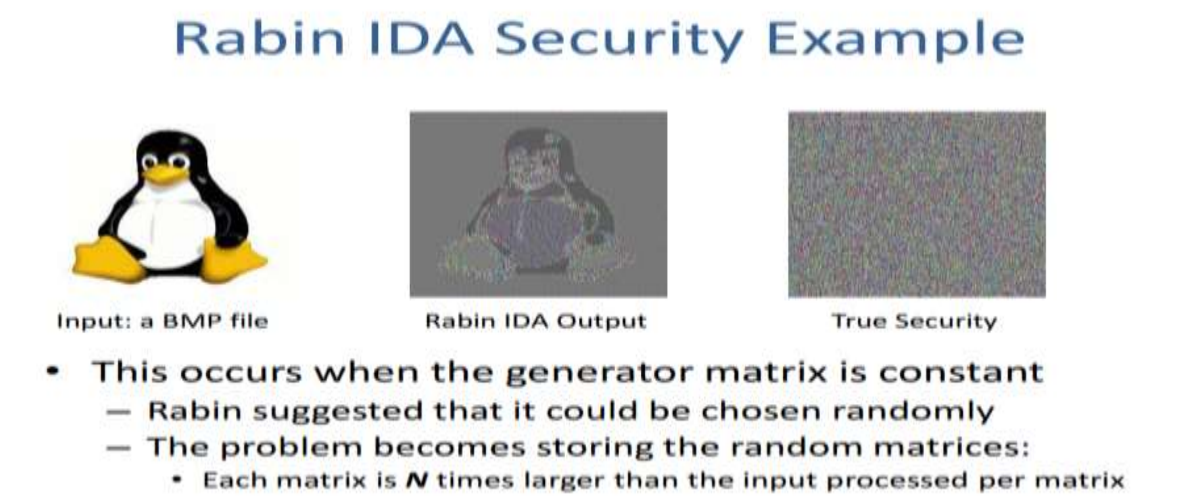
• Raw storage requirements are: (N / K) · Input Size

– Very efficient since (N / K) may be chosen close to 1

• Security of Rabin is not as strong as Shamir

– Having fewer than K shares yields some information

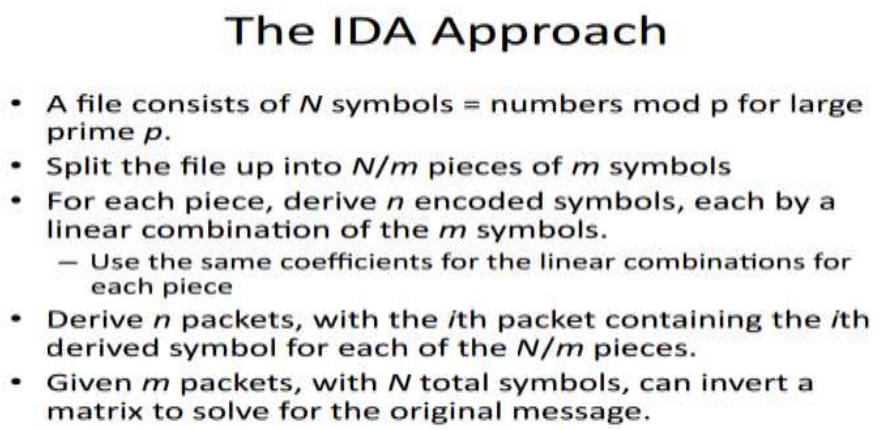
– Repetitions in input create repetitions in output

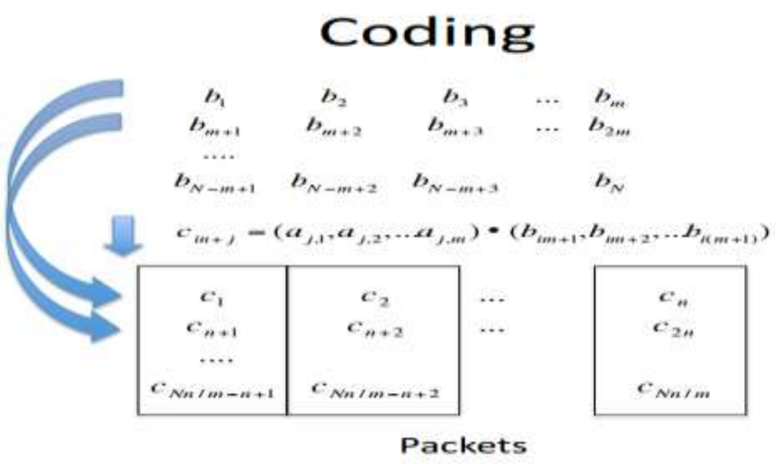


DRAWBACK/POINT OF FAILURE

Shares are typically a complex function of a larger subset of data which in turn affects efficiency

OUTPUTS





Decoding

c_1	c_2	$\cdots$	c_m
c_{n+1}	c_{n+2}	$\cdots$	c_{n+m}
$\cdots$			
$c_{Nn/m-n+1}$	$c_{Nn/m-n+2}$		$c_{Nn/m-n+m}$

For convenience, assume you get first m packets

Let $A = (a_{i,j}), 1 \le i, j \le m$

$$A \bullet \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1} \bullet \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix}$$

Need to be able to invert coefficient matrix, for any set of packets that is received.

Coefficients

- Need the corresponding matrix to be invertible.
- Solution 1: Coefficient chosen according to a Cauchy matrix.
  - Resulting m by m matrix of received coefficients can be inverted in $\Theta(m^2)$ time.
- Solution 2: Vandermonde matrix.
- Solution 3: Choose the coefficients $a_{i,j}$ randomly.
  - Linearly independent with high probability.
  - But $\Theta(m^3)$ decoding by standard means.

Krawczyk’s Computational Secret Sharing (CSS)

EFFICIENCY

Krawczyk’s proposed a non-perfect m-threshold scheme, where m shares recover the secret but m - 1 shares give no information on the secret, in which shares corresponding to a secret set S are of size log |S|/n (where |S| denotes the cardinality of the set S) plus a short piece of information n whose length does not depend on the secret size but just on the security parameter

COMPLEXITY

The CSS scheme becomes inefficient and complex when the size of the access structure is not a polynomial in n

SECURITY

• In 1993, Hugo Krawczyk combined elements of Shamir’s Secret Sharing with Rabin’s IDA
• The SSMS method:
– Input is encrypted with a random encryption key
– Encrypted result is dispersed using Rabin’s IDA
– Random key is dispersed using Shamir’s Secret Sharing
• Yields a computationally secure secret sharing scheme with good security and efficiency
The scheme of Krawczyk is very and combines in a natural way traditional (perfect) secret sharing schemes, encryption, and known information dispersal algorithms. It is provable secure given a secure (private key) encryption function.

DRAWBACKS/POINT OF FAILURE

The time usage for this scheme is dominated by the Information Dispersal algorithms so if this fails then the entire process fails.

OUTPUT

Distribution Scheme of SS:

- Chose a random encryption key $k \in K$. Encrypt the secret $s \in S$ using the encryption function ENC under the key k, let $f = ENC_k(s)$.
- Using IDA partition the encrypted file f into n fragments, $g_1, \ldots, g_n$, and distribute them to the participants in $\mathscr{A}$.
- Using PSS generate n shares for the key k, denoted $v_1, \ldots, v_n$, and distribute them to the participants in $\mathscr{A}$.

The share of each participant P_i, $1 \leqslant i \leqslant n$ consists in $w_i = (g_i, v_i)$.

Reconstruction Scheme of SS:

- Each set of participants A in the access structure collect their shares.
- Using IDA reconstruct f out of the collected values g_i for all $P_i \in A$.
- Using PSS recover the key k out of v_i for all $P_i \in A$.
- Decrypt f using k to recover the secret s.

The following theorem shows that the above scheme is a computational secret sharing scheme.

REFRENCES

http://people.seas.harvard.edu/~salil/rabin2011-slides/rabin2011-mitzenmacher.pdf

https://en.wikipedia.org/wiki/Shamir's_Secret_Sharing

http://airccse.org/journal/ijccsa/papers/2412ijccsa06.pdf

http://kimh.github.io/blog/en/security/protect-your-secret-key-with-shamirs-secret-sharing/