More Versatile Secret Sharing

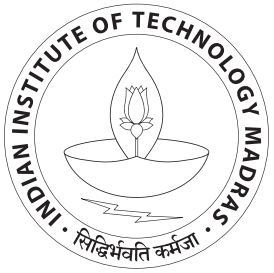
A Project Report

submitted by

MEESA SHIVARAM PRASAD

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2022

THESIS CERTIFICATE

This is to certify that the thesis entitled **More Versatile Secret Sharing**, submitted by **Meesa Shivaram Prasad** (**CS18B056**), to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Aishwarya T
Research Guide
Assistant Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 27th May 2022

ACKNOWLEDGEMENTS

I thank Aishwarya Madam who is my primary source of guidance through out my project and I also want to thank Yadu Vasudev sir for coordinating the B.tech project course.

I would like to thank Raghavendra,Simran (PhD scholars) for helping me out correcting errors in report and also involving in good discussions relating to project.

ABSTRACT

KEYWORDS: Secret Sharing,ADSS

Secret sharing is a method to divide the secret into many shares/parts such that secret can be constructed by authorized set of shares and unauthorized set of shares reveals least possible amount of information. Secret sharing was first proposed by Shamir [1979] in the paper "How to share a secret?".

In classical notion if some shares are corrupted then there is no way to identify the errors or recover from those errors, and we also need the schemes to be deterministic because if in case some shareholders lost their shares re-sharing of the shares will be easy. So [Bellare, Dail, and Rogaway, 2020] augmented the classic notion with privacy when there is imperfect randomness,authenticity,error correction and also made the scheme deterministic and they call it as Adept secret sharing scheme(ADSS)

In this report we present our research in the field of secret sharing. We present our work on reducing the time complexity of Error recovery algorithm of adept secret sharing scheme from non polynomial time to polynomial time for threshold schemes with some assumptions on Adversary.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

SSS	Shamir's secret sharing
ADSS	Adept Secret sharing scheme

NOTATION

$\mathcal{A}$	Access Structure
λ	Security parameter for hash function collision resistance
negl	It's negligible function that's defined in the paper later.
$\mathbb{A}$	Adversery
$\mathbb{I}$	Input selector
$\{0,1\}^{**}$	Set of all list's of strings
$\{0,1\}*$	Set of strings over $\{0,1\}$
$\perp$	Blank symobol indicates nothing is returned
S	Set of shares
$Share*$	Vector of shares
H	Random Oracle hash function
$\mathbb{S}$	General ADSS scheme
$\mathbb{S}1$	Base level shamir's type secret sharing scheme that achieves Priv$ security
$\mathbb{SS}$	ADSS scheme with Auth-security
$Adv^{priv}_{\mathbb{S},\mathbb{I}}(\mathbb{A})$	Privacy advantage of $\mathbb{A}$ relative to $\mathbb{I}$ for scheme $\mathbb{S}$
$Adv^{auth}_{\mathbb{S}}(\mathbb{A})$	Advantage of $\mathbb{A}$ for breaking the Authenticity security for scheme $\mathbb{S}$
$Adv^{errx}_{\mathbb{S}}(\mathbb{A})$	Advantage of $\mathbb{A}$ for breaking the Error security for scheme $\mathbb{S}$
Share	Share generation algorithm
Recover	Message Reconstruction algorithm
Known	Known information to the recovering party
ϵ	Empty string
Priv	Privacy property
Priv$	Weaker security property than Priv
Auth	Authenticity property
Errx	Error correction property
AX	Transformation to get Authenticity property
EX	Transformation to get Error correction property
ES	Efficient error recovery construction that I proposed

CHAPTER 1

Introduction

Secret Sharing is a way to share the secret message by dividing it into many parts such that secret can be reconstructed only when authorized group of people come together for reconstruction. Such schemes are very helpful for storage of secure information such as cryptographic keys. For example, one might think of protecting a private key by placing it in a secure location, but if this secure location is compromised or damaged we lose the key. To avoid this, we may think of making multiple copies of the key but this increases risk of exposing the key.

So the solution provided by secret sharing is to divide the secret key into many shares and define that if some authorized set of people or shares come together then the secret key can be reconstructed and an unauthorized subset of people can't reconstruct the secret.

Access structure:

Access structure is a way to define which set of parties are authorised to reconstruct the secret. Let's consider a scheme in which out n people if any t or more people come together then the reconstruct the secret, we can represent this by (k, n) and these are called threshold access structures. But in general access structure is a set of set of positive numbers, where any set of people that belongs to this $\mathcal{A}$ access structure are called authorized , if not then they are called un-authorized.

Secret Sharing:

We can view secret sharing as pair of $(Share, Recover)$ algorithms along with an

Access structure $\mathcal{A}$. The Share algorithm on taking message and $\mathcal{A}$ as input will generate shares and Recover algorithm on input some subset of shares aims reconstructing the original secret.

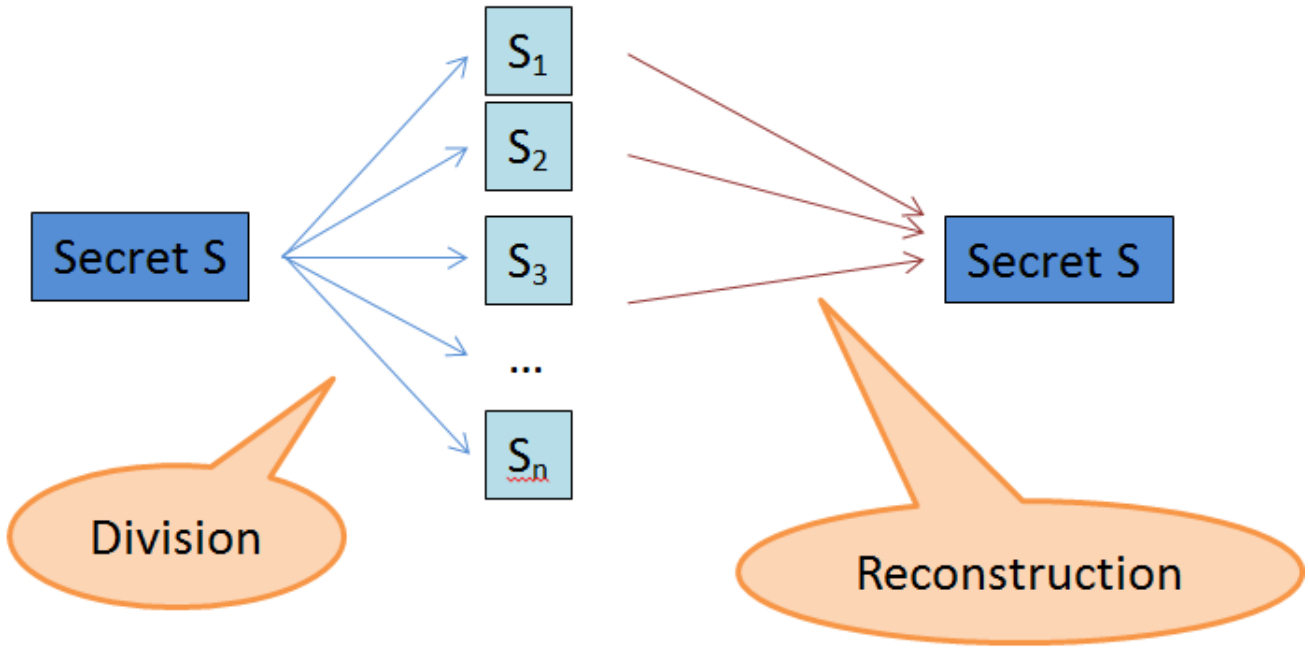


Figure 1.1: Illustration of Secret sharing

The recovery algorithm should reconstruct the secret when authorized set of shares come for reconstruction, and also un-authorized set of shares should reveal the least possible amount of information.

Shamir's Secret Sharing Scheme:

Shamir [1979] described a threshold secret sharing scheme, where if more than $t-1$ people come together then the secret can be reconstructed, else we can't. The main idea of this scheme is based on polynomial interpolation algorithm, i.e., if we have atleast t points then we can uniquely construct the $(t-1)^{th}$ degree polynomial.

The scheme is as follows, choose a polynomial of degree $(t-1)$ with coefficients

$a_1, a_2, \ldots, a_{t-1}$ chosen at random , and a_0 is secret message, the shares are $(i, f(i))$ where $i \in [1 \ldots n]$. Now if atleast t people come together we can uniquely construct the polynomial and find the secret, but if less than t people there can be infinitely many possibilities so we can't reconstruct the secret.

Shamir's secret sharing scheme satisfies the basic notion of security but will be having problems when few of those shares got corrupted, and as the scheme is not deterministic we should re-share all the shares if some share holder lost his/her share.

Adept Secret Sharing Scheme:

[Bellare, Dail, and Rogaway, 2020] proposed ADSS scheme which is a deterministic secret sharing scheme. They have provided the notion of privacy even if there is imperfect randomness, and authenticity notion (i.e., each share contributes to exactly one message) and also Error correction which means recover the message if it is recoverable despite there are few shares that got corrupted.

We worked on open problem of this Bellare *et al.* [2020] paper which is improving the time complexity of error recovery algorithm from non-polynomial to polynomial in number of shares.

Organization:

This report is divided into 3 chapters, where first chapter is Introduction, the chapter-2 explains the preliminaries, Shamir's secret sharing scheme, Adept Secret Sharing scheme. We present our improvements on ADSS paper in chapter-3.

CHAPTER 2

Preliminaries and Related Work

2.1 Preliminaries

Definition 2.1.1 (**Access Structures** (adapted from Bellare *et al.* [2020])). *If we number parties* $1, 2 \ldots, n$ *then Access structure* $\mathcal{A}$ *is set of sets of the positive numbers. Let* $n(\mathcal{A})$ *be the number of parties in* $\mathcal{A}$*, then we say set* $U \subseteq [1 \ldots n(\mathcal{A})]$ *is authorized if* $U \in \mathcal{A}$ *and unauthorized if* $U \notin \mathcal{A}$

Definition 2.1.2 (**Threshold access structures** (adapted from Bellare *et al.* [2020])). *let* $1 \leq t \leq n$*, the threshold access structure* $\mathcal{A}_{n,t}$ *is* $\{U \subseteq [1 \ldots n(\mathcal{A})] : |U| \geq t\}$

Classical secret sharing can be formalized as $\mathbb{S} = (Share, Recover)$ along with an Access Structure $\mathcal{A}$. Basically a pair of $(Share, Recover)$ algorithms along with Access structure $\mathcal{A}$.

Definition 2.1.3 (**Share** (adapted from Bellare *et al.* [2020])). *It takes in secret message* M *, access structure* $\mathcal{A}$ *as input and probabilistically outputs* $n = n(\mathcal{A})$ *shares which is list of strings,* $\boldsymbol{S} \leftarrow Share(M, \mathcal{A})$

Definition 2.1.4 (**Recover** (adapted from Bellare *et al.* [2020]). *Given input list of strings, with each element either* $\diamond$ *or string (Share) , where* $\diamond$ *represents missing share. The algorithm reconstructs the secret message M using the shares that are present.* $\boldsymbol{M} \leftarrow Recover(S)$

Definition 2.1.5 (**Correctness** (adapted from Bellare *et al.* [2020])**.** *Let* $\mathbf{S} \leftarrow Share(M)$ *, and* $S = (S_1, S_2, .., S_n)$ *and* $U \subseteq [1 \dots n]$ *, then let* S_U *be the sub-vector vector of size n with* i^{th} *component* S_i *if* $i \in U$ *else* $\diamond$ *then we need* $Recover(\mathbf{S}_U) = M$ *if* $U \in \mathcal{A}$*.*

Definition 2.1.6 (**Classical Security Notion** (adapted from Bellare *et al.* [2020])**.** *The security notion of classical secret sharing requires that unauthorized sub-vector of shares should reveal least possible amount of information , i.e., let* M_1, M_2 *be any two messages and* $U \notin \mathcal{A}$*, then the distribuitons* $(Share(M_1))_U$ *and* $(Share(M_2))_U$ *should be identical.*

2.2 Shamir's Secret Sharing (SSS)

This secret sharing scheme was introduced by [Shamir, 1979] , and it's a threshold secret sharing scheme, here the secret can be reconstructed only if atleast k shares are available out of n, else we can't reconstruct the share. Formally access structure $\mathcal{A}_{n,k}$ is $\{U \subseteq [1 \dots n(\mathcal{A})] : |U| \geq k\}$

The main idea of this scheme is based on Lagrange interpolation theorem. We can find the polynomial of degree less than k, with k points uniquely. So the Share algorithm on input secret a_0 chooses $a_1, a_2, ...a_{k-1}$ randomly and constructs k^{th} degree random polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$, and then make $n \geq k$ shares $S_i =< i, f(i) >$.

Recover algorithm on receiving input at least k shares, if we consider some subset of k shares out of it we can obtain the secret a_0 by interpolation algorithm as follows.

$$a_0 = \sum_{j=0}^{j=k-1} y_j \prod_{i=0, i \neq j}^{i=k-1} \frac{x_i}{(x_i - x_j)}$$

Shamir's secret sharing guarantees us that unauthorized subset of shares reveal least possible information, in the above case if we have less than k points there are infinitely many possible $(k-1)^{th}$ degree polynomials.

Problems of using Shamir's secret sharing:

- The scheme is not deterministic, even if one share is lost by share holder, we have to run the share algorithm again to distribute shares to each and every share holders.
- If some of the shares got corrupted then this approach outputs some random message, so here Adversary can control the output of message by changing few shares. There is no authenticity guarantee we have in this scheme
- If we get some random message as output, then we know some shares got corrupted and in this scheme there is no way to identify which shares are corrupt, which are not and recover the message if it's recoverable.

2.3 Adept Secret Sharing Scheme (ADSS)

2.3.1 Goals

Bellare,Dail,and Rogaway identified some gaps in the definitions of classical secret sharing and felt the following goals were required. (1) We need secret sharing schemes to be deterministic, because with the help of deterministic schemes we can do share regeneration easily if some share holder's loose their shares. (2) We need to provide privacy even if the dealer doesn't employee fresh coins or imperfect randomness, (3) We need to provide authenticity that is each share should contribute to at max 1 message, and also (4) We need to provide error correction, that is recover the message if it is recoverable i.e., when few or no shares got corrupted we should be able to recover the message.

This section explains the paper on ADSS scheme by [Bellare, Dail, and Rogaway, 2020] which fulfills the above mentioned goals, and in the next chapter we discuss on improvements that we done over this paper.

2.3.2 ADSS Syntax

Adept Secret Sharing scheme (ADSS) is defined as tuple of three deterministic functions *Acc*,Share,Recover

Definition 2.3.1 (**Acc**: Bellare *et al.* [2020])**.** *This function is used to associate access structures* $A = Acc(\mathcal{A})$ *with all strings* $\mathcal{A} \in Access$*.*

$$Acc : Access \rightarrow AS$$

Here $Access = \{0,1\}^*$ *and AS is set of all possible access structures.*

Definition 2.3.2 (**Share**: Bellare *et al.* [2020])**.** *It is a Share generation algorithm*

$$Share:\ Access \times Msg \times Rand \times Tag \rightarrow Share^*$$

It takes in a description $\mathcal{A} \in Access$, secret message $M \in Msg$, random coins $R \in Rand$, associated data $T \in Tag$ and outputs vector of shares. Here sharing algorithm will generate appropriate number of shares for the given access structure, i.e., $|Share(\mathcal{A}, M, R, T)| = n(Acc(\mathcal{A}))$

Definition 2.3.3 (**Recover** Bellare *et al.* [2020])**.** *Message reconstruction algorithm*

$$Recover:\ Shares \rightarrow Msg \times Shares \cup \{\perp\}$$

Recover algorithm receives input a set of shares and using some subset of these shares if it can output a message , then it outputs the pair (Msg, V) where

$V \subseteq Shares$ that are used for message reconstruction, if it can't output the message then $\perp$ symbol is returned.

Random Oracles Bellare *et al.* [2020]):

Both the Share and Recover algorithms are allowed to call a random oracle Hash $H \in \Omega$, where Ω is a set of functions H: $\mathbb{N} \times \{0,1\}^{**} \rightarrow \{0,1\}^{*}$ such that $|H(l, \mathbf{X})| = l$, here $\{0,1\}^{**}$ means list of strings. During the entire discussion we indicate this Hash function H by keeping it as a superscript to Share and Recover Algorithms , i.e., $Share^{H}$ and $Recover^{H}$

Definition 2.3.4 (Basic Correctness Bellare *et al.* [2020])**.** *An ADSS scheme* $\mathbb{S} = (Acc, Share, Recover)$ *satisifies basic correctness if for every* $\mathcal{A} \in Access, M \in Msg, R \in Rand, T \in Tag, H \in \Omega, S \leftarrow Share^{H}(\mathcal{A}, M, R, T)$ *and* $U \subseteq [1 \ldots n(Acc(\mathcal{A}))]$ *if* $U \in Acc(\mathcal{A})$ *then* $Recover^{H}(S[U]) = (M, S[U])$, *else if* $U \notin Acc(\mathcal{A})$ *then* $Recover^{H}(S[U]) = \perp$

Full Correctness Bellare *et al.* [2020]):

A small change in the recover algorithm output in contrast to the previous discussion, instead of outputting only (M, V) we want the recover algorithm to output (M, R, V) where R is the randomness that has been used to generate shares, now with this setting we say ADSS scheme is fully correct if it satisfies Basic Correctness and also Validity. Here, validity means we don't want the recover algorithm to output random (M, R, V) in other words the recover doesn't lie, we can always check Validity by running $Share^{H}(V.access, M, R, V.tag)$ and see whether V is an authorized subset of shares generated by the algorithm or not.

2.3.3 Security Definitions

In this sub-section we discuss the security definitions of the ADSS scheme and in later sections we propose the schemes that satisfies these security definitions.

Definition 2.3.5 (**Privacy** Bellare *et al.* [2020])**.** *The privacy advantage of* $\mathbb{A}$ *relative to* $\mathbb{I}$ *is defined as*

$$Adv^{priv}_{\mathbb{S},\mathbb{I}}(\mathbb{A}) = 2Pr[G^{priv}_{\mathbb{S},\mathbb{I}}(\mathbb{A})] - 1$$

The privacy security is captured by the probability of an Adversary winning the below game. The game has one helper function called Deal oracle which is the next procedure.

Game $G^{priv}_{\mathbb{S},\mathbb{I}}(\mathbb{A})$

procedure $Main()$
$c \leftarrow \{0,1\}$; $H \leftarrow \Omega$
$q \leftarrow 0$; $(St, \mathbf{B}) \leftarrow \mathbb{I}^{Deal}$
if $(\exists j : \mathbf{B}[j] \in Acc(\mathbf{A}[j]))$ then return false
$c^{'} \leftarrow \mathbb{A}^{H}(St, \mathbf{S}_1[\mathbf{B}[1]], ..., \mathbf{S}_q[\mathbf{B}[q]], \mathbf{P})$
return $c = c^{'}$

procedure $Deal(A, M_0, M_1, R, T)$
$q \leftarrow q + 1$; $\mathbf{A}[q] \leftarrow A$
$\mathbf{S}_q \leftarrow \mathbb{S}.Share^{H}(A, M_c, R, T)$; $\mathbf{P}[q] \leftarrow S_q.pub$
return

- Game chooses bit c randomly
- Input Selector ($\mathbb{I}$) (Dealer) can query the deal oracle with A, M_0, M_1, R, T of his choice for q times
- Shares are generated for A, M_c, R, T and nothing is returned on query to Deal
- The queries to the Deal oracle by $\mathbb{I}$ should be non repetitive, else one can trivially discover the bit c.
- $\mathbb{I}$ now chooses some unauthorized subset of parties corresponding to each query

- If $\mathbb{I}$ chooses some authorized subset of parties then line no 4 return false
- Now these unauthorized subset of shares, and also public portions are shared to adversary $\mathbb{A}$ and it should predict the bit c
- Adversary($\mathbb{A}$) wins the game if it correctly predicts the bit c.

We will be getting privacy only when $(M_0, R), (M_1, R)$ queries of the Input Selector $\mathbb{I}$ are unpredictable.So to define the predictability of $\mathbb{I}$ the below game has been defined.

Game $G_{\mathbb{I}}^{pred}(\mathbb{P})$

1: procedure $Main()$
2: $q \leftarrow 0; (St, \mathbf{B}) \leftarrow \mathbb{I}^{Deal}$
3: $(M, R) \leftarrow \mathbb{P}(\mathbf{A}, \mathbf{B}, \mathbf{T}, \mathbf{L}, St)$
4: return $((M, R) \in D)$

1: procedure $Deal(A, M_0, M_1, R, T)$
2: $D \leftarrow D \cup \{(M_0, R), (M_1, R)\}$
3: $q \leftarrow q + 1; \mathbf{A}[q] \leftarrow A;$
4: $\mathbf{R}[q] \leftarrow R; \mathbf{T}[q] \leftarrow T; \mathbf{L}[q] \leftarrow |M_0|$
5: return

$$Adv_{\mathbb{I}}^{pred}(\mathbb{P}) = Pr[G_{\mathbb{I}}^{pred}(\mathbb{P})] \text{ and } pred(\mathbb{I}) = max_{\mathbb{P}}\{Adv_{\mathbb{I}}^{pred}(\mathbb{P})\}$$

- $\mathbb{I}$ queries *Deal* for q times and *Deal* just store the queries $(M_0, R), (M_1, R)$ pairs.
- Here for the second stage we leak the information like which parties are corrupted,length of message,tag etc
- Now adversary $\mathbb{P}$ should predict a (M, R) pair that $\mathbb{I}$nput selector has queried
- We say $\mathbb{I}$ is unpredictable if $pred(\mathbb{I})$ is negligible.

Let's define new class of Input selectors called $\mathbb{I}^{priv\$}$. where these input selectors call $Deal^*$ subroutine which is defined as on input (A, M_0, M_1, R, T) , R^* is chosen at random from $\{0,1\}^r$ and we call $Deal(A, M_0, M_1, R^*, T)$. It's easy to see $pred(\mathbb{I}^{priv\$}) \leq q.2^{-r}$. When we use these type of input selectors we will write $Adv_{\mathbb{S},\mathbb{I}}^{priv\$}(\mathbb{A})$ instead of $Adv_{\mathbb{S},\mathbb{I}}^{priv}(\mathbb{A})$

Definition 2.3.6 (**Authenticity** Bellare *et al.* [2020])**.** *The Authenticity advantage of* $\mathbb{A}$ *with respect to S is defined as*

$$Adv^{auth}_{\mathbb{S}}(\mathbb{A}) = Pr[G^{auth}_{\mathbb{S}}(\mathbb{A})]$$

Game $G^{auth}_{\mathbb{S}}(\mathbb{A})$

$H \leftarrow \Omega$
$(S, S^{'}) \leftarrow \mathbb{A}^{H}$
$(M, V) \leftarrow \mathbb{S}.Recover^{H}(S)$
$(M^{'}, V^{'}) \leftarrow \mathbb{S}.Recover^{H}(S^{'})$
return $V \cap V^{'} \neq \Phi$ and $M \neq M^{'}$

- Adversary can choose any two set of shares $(S, S^{'})$
- If Recovery outputs two different messages such that there is one share that is contributing to 2 different messages then $\mathbb{A}$ wins.
- Essentially telling whether it's any share good or bogus it has to contribute to exactly one message

Definition 2.3.7 (**Errx Security** Bellare *et al.* [2020])**.** *The Errx Security for an ADSS scheme is defined as*

$$Adv^{errx}_{\mathbb{S}}(\mathbb{A}) = Pr[G^{errx}_{\mathbb{S}}(\mathbb{A})]$$

Game $G^{errx}_{\mathbb{S}}(\mathbb{A})$

$H \leftarrow \Omega$; $(K, S) \leftarrow \mathbb{A}^{H}$
return $\mathbb{S}.Recover^{H}(K, S) \neq UniqueExplanation^{H}(K, S)$

procedure $UniqueExplanation^{H}(K, S)$
if $\exists (A, M, R, V) \in Explanations^{H}$ such that $(\hat{A}, \hat{M}, \hat{R}, \hat{V}) \in Explanations^{H} \implies (A = \hat{A} \cap M = \hat{M} \cap R = \hat{R} \cap \hat{V} \subseteq V)$ **then**
    return this (necessarily unique) (M, R, V)
else
    return $\perp$
end if

procedure $Explanations^{H}(K, S)$
return $\{(V.as, M, R, V) : \hat{S} \subseteq S, (M, R, V) = \mathbb{S}.Recover^{H}(K, \hat{S})\}$

- A small change in the syntax of outputs and inputs of recover algorithms.
- We allow recovery party having known information $K \in Access \cup Shares$
- We want the Recover to output R along with the secret message M
- $Adv^{errx}_{\mathbb{S}}(\mathbb{A}) = Pr[G^{errx}_{\mathbb{S}}(\mathbb{A})]$, and perfect error correction means $Adv^{errx}_{\mathbb{S}}(\mathbb{A}) = 0$.
- The adversary wins in the above game if it forces the recover something other than the UniqueExplanation.
- Explanations means a set of tuples $\{(V.as, M, R, V) : \hat{S} \subseteq S, (M, R, V) = \mathbb{S}.Recover^{H}(K, \hat{S})\}$, and UniqueExplanations means if there is exist some explanation such that all $\hat{V}'s$ of other explanations are subset of V and all explanations having same A, M, R.
- If there exists such an unique explanation the UniqueExplanations function return (M, R, V) else $\perp$

2.3.4 ADSS Constructions

Base level Scheme $\mathbb{S}1$Bellare *et al.* [2020]:

procedure $\mathbb{S}1.Share(A, M, R, T)$

$(k, n) \leftarrow A$
$M_1 || \ldots || M_m \leftarrow M$ where $|M_i| = \beta$
for $(i, j) \in [1 \ldots (k-1)] \times [1 \ldots m]$ **do**
    $R_{j,i} \leftarrow f^{\beta}_{R}(i, j)$
end for
for $i \in [1 \ldots n]$ **do**
    for $j \in [1 \ldots m]$ **do**
        $B_{i,j} \leftarrow M_j + R_{j,1}.i + R_{j,2}.i^2 + \cdots + R_{j,k-1}.i^{k-1}$
    end for
    $S_i \leftarrow (i, (k, n), B_{i,1} \ldots B_{i,m}, \epsilon, \epsilon)$
end for
return $(S_1, S_2, \ldots, S_n)$

Theorem 1. *Let* $S_1 = S_1[\beta, f]$ *with* $\beta \geq 2$ *and* $f : \{0,1\}^k \times \mathbb{N} \times \{0,1\}^{**} \rightarrow \{0,1\}^*$. *Then* S_1 *satisfies Priv\$. Given input-selector* $I \in I^{priv\$}$ *and Priv-adversary* $\mathbb{A}$ *we build PRF-adversary* $\mathbb{B}$ *such that* $Adv^{priv\$}_{\mathbb{S}1,\mathbb{I}}(\mathbb{A}) \leq Adv^{prf}_{f}(\mathbb{B})$

procedure $\mathbb{S}1.Recover(S)$

$t \leftarrow |S|$; $\{S_1, \ldots S_t\} \leftarrow S$
for $i \in [1 \ldots t]$ **do**
  $(l_i, (k_i, n_i), B_{i,1} \ldots B_{i,m_i}, \epsilon, \epsilon) \leftarrow S_i$
end for
$(k, n, m) \leftarrow (k_1, n_1, m_1)$
if $t < k$ **then**
  return $\perp$
end if
for $j \in [1 \cdots]$ **do**
  $\Psi_j(x) \leftarrow Interpoplate_\beta(\{(l_1, B_{1,j}) \ldots (l_k, B_{k,j})\})$
  $M_j \leftarrow \Psi_j(0)$
end for
return $(M_1 \ldots M_m, S)$

This base level scheme $S1$ is essentially Shamir's secret sharing scheme, in the share algorithm initially the message is split into blocks of size β to form m blocks of the message. Now random coefficients are generated using pseudorandom function with key input randomness R.

Now the polynomial will be constructed using these Coefficients with constant as message block, and we evaluate $f(i)$ and store it in the share. Recover algorithm on receiving these $(i, f(i))$ runs Lagrange interpolation algorithm to obtain the polynomial f and calculates the message by $f(0)$.

AX scheme Bellare *et al.* [2020]:

Theorem 2. *Let* $\mathbb{SS} = AX[\mathbb{S}, f]$ *where* $\mathbb{S}$ *is an ADSS scheme with* $f : \{0,1\}^k \times \mathbb{N} \times \{0,1\}^{**} \rightarrow \{0,1\}^*$ *then*

- *If* $\mathbb{S}$ *is Priv\$-secure then* $\mathbb{SS}$ *is Priv-Secure.*
- $\mathbb{SS}$ *is Auth-secure.*

In the AX transformation we assume $f : \{0,1\}^k \times \mathbb{N} \times \{0,1\}^{**} \rightarrow \{0,1\}^*$. is a PRF, and

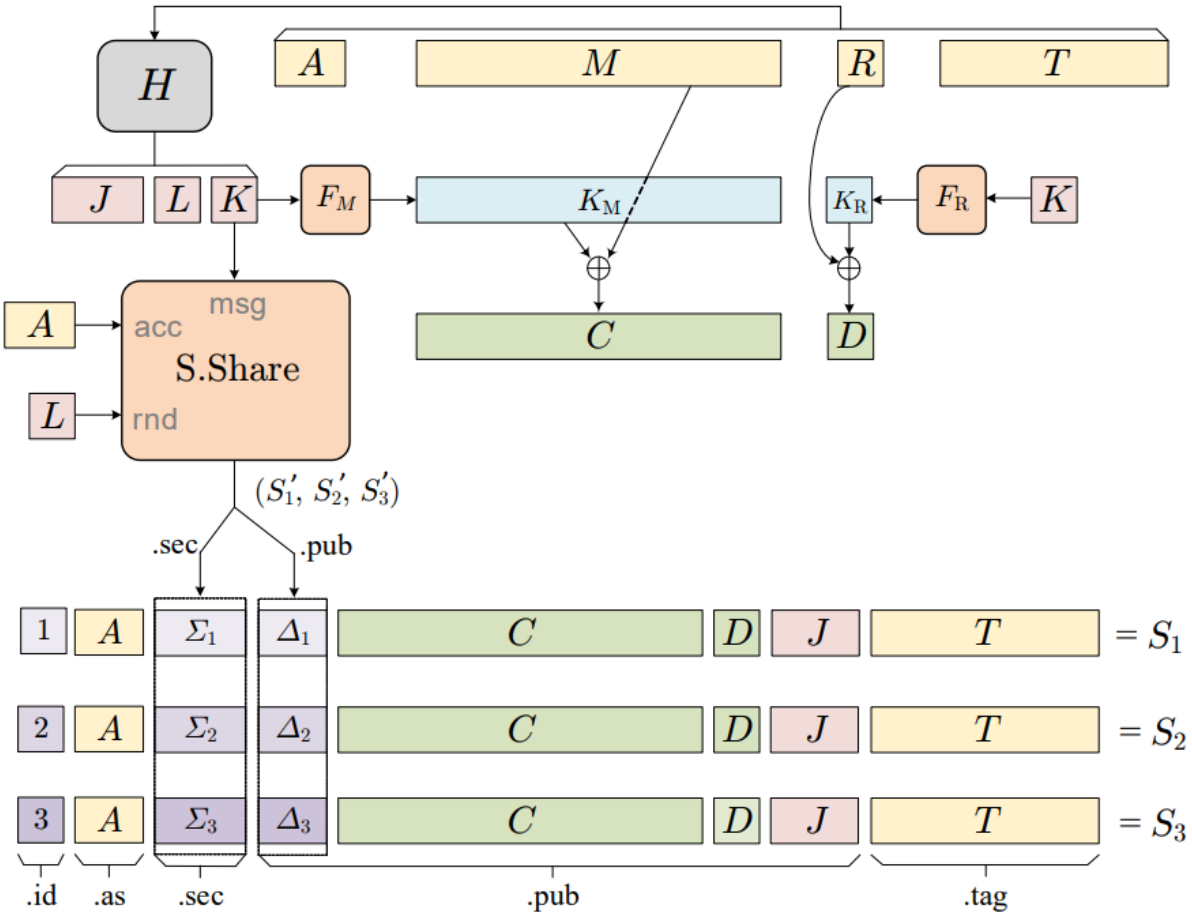


Figure 2.1: Pictorial representation of AX construction taken from 2

procedure $\mathbb{SS}.Share^{Hash}(A, M, R, T)$

$J||K||L \leftarrow H^{4\kappa}(A, M, R, T)$ where $|J| = 2\kappa$ and $|K| = |L| = \kappa$
$C \leftarrow M \oplus f_K^{|M|}(\epsilon)$
$D \leftarrow R \oplus f_K^{\kappa}(0)$
$n \leftarrow n(\mathbb{S}.Acc(A))$
$(S'_1, \ldots, S'_n) \leftarrow \mathbb{S}.Share^h(A, K, L, \epsilon)$
for $i \in [1 \ldots n]$ **do**
    $\Sigma_i \leftarrow S'_i.sec; \Delta_i \leftarrow S'_i.pub$
end for
for $i \in [1 \ldots n]$ **do**
    $P_i \leftarrow (\Delta_i, C, D, J)$
    $S_i \leftarrow (i, A, \Sigma_i, P_i, T)$
end for
return $(S_1, S_2, \ldots, S_n)$

procedure $h^l(x)$
return $Hash(l, 0||x)$

procedure $H^l(x)$
return $Hash(l, 1||x)$

procedure $\mathbb{SS}.Recover^{Hash}(K, S)$

$t \leftarrow |S|$
$\{S_1, \ldots, S_t\} \leftarrow S$
for $i \in [1 \ldots t]$ **do**
    $(j_i, A_i, \Sigma_i, P_i, T_i) \leftarrow S_i$
    $(\Delta_i, C_i, D_i, J_i) \leftarrow P_i$
    $S'_i \leftarrow (j_i, A_i, \Sigma_i, \Delta_i, \epsilon)$
end for
$S' \leftarrow \{S'_i : i \in [1 \ldots t]\}$
$(K, G) \leftarrow \mathbb{S}.Recover^h(S')$
$(., A, ., (., C, D, J), T) \leftarrow S_1$
$M \leftarrow C \oplus f_K^{|C|}(\epsilon)$
$R \leftarrow D \oplus f_K^{\kappa}(0)$
if $H^{4\kappa}(A, M, R, T) = J||K$ and $S.id \in S.Acc(A)$ and $S \subseteq \mathbb{SS}.Share^{Hash}(A, M, R, T)$ **then**
    return (M, R, S)
end if
return $\perp$

both share,recover algorithms have access to random oracle Hash function.Now proof of security for the theorem-2 depends on prf security as well as hash collision resistance security.

A quick proof for Authenticity is as follows, in the recover algorithm of $\mathbb{SS}$ we want all the shares have same J value.Now if a share can contribute to two different messages then it must be the case that Hash value of those two messages should collide same for the first 2κ bits but that happens at $negl(\lambda)$ probability as we assume hash function is collision resistance, that concludes the proof.

EX scheme Bellare *et al.* [2020]):

procedure $\mathbb{SS}.Recover(K, S)$

let $S_1, \ldots S_w \in P(S)$ including all K-Plausible sets of shares
$S_i \subseteq S_j \implies i \geq j$
for $i \in [1 \ldots w]$ **do**
  if $(M, R, V) \leftarrow \mathbb{S}.Recover(K, S_i)$ and $V = S_i$ **then**
    goto 8
  end if
end for
return $\perp$
$\{S_1^{'}, \ldots, S_u^{'}\} \leftarrow \{S_{i+1}, \ldots, S_w\} \backslash P(V)$
for $i \in [1 \ldots u]$ **do**
  if $(M^{'}, R^{'}, V^{'}) \leftarrow \mathbb{S}.Recover(K, S_i^{'})$ and $V^{'} \not\subset V$ **then**
    return $\perp$
  end if
end for
return (M, R, V)

Theorem 3. *Let* $\mathbb{SS} = EX[\mathbb{S}]$ *with* $\mathbb{S}$ *be any ADSS scheme with full correctness then If* $\mathbb{S}$ *is auth-secure then so is* $\mathbb{SS}$, $\mathbb{SS}$ *is perfectly Errx-secure.* $Adv_{\mathbb{SS}}^{errx}(\mathbb{A}) = 0$

- P means Power set, set of shares are K plausible if all share same access structure or some other criterion we can keep.

- We try to find one explanation in first for loop In second loop we try to find another possible explanation if we find one more then we return $\perp$ else returns (M, R, V)
- In worst case the time complexity of this recover function is exponential in number of input shares.

CHAPTER 3

Efficient Error Recovery Construction

This chapter discusses the efficient error recovery construction that I designed, which runs in polynomial time, unlike the previous EX construction 2.3.4 that run in exponential time.

We have made a few assumptions on adversary $\mathbb{A}$, and we deal here only with threshold schemes. Let us say the threshold is t.
Assumptions on adversary:

- Adversary $\mathbb{A}$ can infiltrate/corrupt atmax $t-1$ shares.
- We also assume that the Recover algorithm knows the Access Structure or, in this case, the threshold t.

We assume Adversary $\mathbb{A}$ cannot see all t shares because then privacy itself will be lost, so $\mathbb{A}$ can corrupt at max $t-1$ shares , and if we don't allow Recover algorithm to know Access structure a simple attack is that the adversary corrupts a few shares and keeps the threshold as 1 in the Access structure field of the shares tag. This attack could make the Error recovery process return $\perp$ because of multiple explanations.

With the above assumptions on adversary ,I propose the following $\mathbb{ES}$ construction, and claim that $\mathbb{ES}$ always produce unique explanations for any input of shares, moreover if the input has good shares greater than threshold then we reconstruct the original secret (M, R) else we return $\perp$.

3.1 $\mathbb{ES}$ Construction

We propose the following construction which is $\mathbb{ES} = EX[\mathbb{SS}]$, where $\mathbb{SS}$ is any enriched ADSS scheme with full correctness. In the sharing algorithm, we initially

procedure $\mathbb{ES}.Share^H(\mathcal{A}, M, R, T)$

1: $S \leftarrow \mathbb{SS}.Share^H(\mathcal{A}, M, R, T)$
2: $n \leftarrow n(Acc(\mathcal{A}))$
3: $(S_1, \ldots S_n) \leftarrow S$
4: **for** $i \in [1 \ldots n]$ **do**
5: $\quad S_i^{'} \leftarrow S_i || H(S_1) || H(S_2) || \ldots || H(S_n)$
6: **end for**
7: return $(S_1^{'}, S_2^{'}, \ldots, S_n^{'})$

run the Share algorithm of $\mathbb{SS}$ to get the shares, and then for every share S_i, we concatenate the hash value of all the shares $H(S_1)||H(S_2)\ldots||H(S_n)$ and call it as $S_i^{'}$ now this vector $S^{'}$ will be the shares outputted by this share algorithm.

In the recover algorithm, at the start, we will filter the shares with the same access structure field known to the recoverer. Later, we extract each share's actual share numbers fields in a_i. We run a for loop of size t. In each iteration, we check according to Hash values maintained by the current share $S_i^{'}$ which all shares are good, i.e., not corrupted. We store this inset V, and now we pass this V to the procedure filter_shares.

filter_shares checks whether all the shares agree with each other that they are not corrupted. So to achieve that, we do the following, We iterate over each share in V, in line number 2, and now check whether all elements in set V are good or not by comparing the hash values maintained by the current share S_i. If any comparison fails, then we return {} (in line 5) empty set as all the shares in V do not agree with each other. Else we extract the first field from each share the actual

procedure $\mathbb{ES}.Recover^{H}(K, S)$

K is the known information which is the threshold t.
$S \leftarrow \{x : x \in S \text{ and } x.access = K\}$
$n \leftarrow |S|$;
for $i \in [1..n]$ **do**
  $(a_i, S_i^{'}) \leftarrow S_i$
end for
$X = \{\}$
for $i \in [1 \ldots n]$ **do**
  $V = \{\}$
  for $j \in [1 \ldots n]$ **do**
    if $S_i^{'}[a_j] = H(S_j^{'}[0])$ **then**
      V.add($\{a_j, S_j^{'}\}$)
    end if
  end for
  $V' \leftarrow filtrate\_shares(V)$
  if $|X| \leq |V'|$ and $|V'| \geq t$ where t is threshold of the access structure. **then**
    $X \leftarrow V'$
  end if
end for
$(M, R, D) = \mathbb{SS}.Recover^{H}(K, X)$
if $(M, R, D) \neq \perp$ **then**
  $G \leftarrow concat\_hash\_values(D, S)$
  return (M, R, G)
end if
return $\perp$

procedure filtrate_shares(V):
for $\{a_i, S_i\} \in V$ **do**
  for $\{a_j, S_j\} \in V$ **do**
    if $S_i[a_j] \neq H(S_j[0])$ **then**
      return {}
    end if
  end for
end for
$V' \leftarrow \{\}$
for $\{a_i, S_i\} \in V$ **do**
  $V'.add(S_i[0])$
end for
return V'

share message and make a set of shares V' and return it.

We maintain final_set X , which will be updated with V' if $|X| \leq |V'|$ and $|V'| \geq t$ where t is threshold of the access structure. If X is Φ empty, then we return $\perp$, else we pass it to $\mathbb{SS}.Recover(A, X)$ and get the output (M, R, D). Now if $(M, R, D) \neq \perp$ we convert D into G by concatenating the respective hash_value_fields of each share and output (M, R, G), else we output $\perp$.

In brief the Share algorithm is providing knowledge about all the shares whether they are correct or not by adding hash values of all shares. In Recover algorithm we iterate over each shares and check in the context of current share which all are good and which are corrupted. So let's say if we are checking in the context of good shares then we'll be getting all good shares in V, but if the current share is corrupted/bad then the set V can contain mix of good and bad, so then we pass this V set through filter_shares function now if V contain a mix of good and bad shares we will be returning {} because the good shares can't verify corrupted shares. And if the set $V$is only full of good shares or full of bad shares then we get back the same V as output. Now as we assumed the number of corrupted shares can't cross t, so we use this condition in if statement to avoid the bad shares and always fill the X with good shares.

Definition 3.1.1 (*negl*)**.** *We say a function $f(n)$ is* negligible *if it is $O(n^{-c})$ for all $c > 0$, and we use $negl(n)$ to denote a negligible function of n.*

Theorem 4. *Let $\mathbb{ES} = EX[\mathbb{SS}]$ with $\mathbb{SS}$ be any enriched ADSS scheme with full correctness, then*

1. *If $\mathbb{SS}$ is auth-secure then $\mathbb{ES}$ is auth-secure.*
2. $Adv^{errx}_{\mathbb{ES}}(\mathbb{A}) \leq negl(\lambda)$

We prove this theorem 4, using the following claim 5.

Claim 5. *Let $\mathbb{ES}$ be ADSS scheme with threshold t, and we assume Adversary can corrupt a maximum of t-1 shares,and H is a collision resistant hash function.*

Consider the contents of set X in $\mathbb{ES}.Recover^H(K, S)$ after reaching line 20.

- *Number of good shares in $S < t \implies Pr[X \neq \{\}] \leq negl(\lambda)$*
- *Number of good shares in $S \geq t \implies Pr[X \neq G] \leq negl(\lambda)$, where G is set of all good shares in S.*

Proof:

Let the input to $\mathbb{ES}.Recover^H$ be (K, S) , and t be the threshold that we got to know from K the known information or access structure. Then let's say $S' \subseteq S$ be the set of good shares, in the for loop at line number 8

- Case 1:

  If the current share S'_i that we are iterating is good share then the set V will be consists of all good shares, and it doesn't include any bad share, for bad share inclusion to happen there must be hash collision which happens at negligible probability.

  Since this set V is containing all good shares so when it has passed through the filter_shares function it won't return {} as everyone is not corrupted their hash values will be verified and so all shares agree each other that they are not corrupted. So in this case the X will be updated with V which is all good shares, only if $|V| \geq t$, where t is threshold.

- Case 2:

  If the current share S'_i that we are iterating is corrupted, then the set V can be consisting of mix of good shares and bad shares.

  - Case 2 a):

    Let say set V contain only bad shares, then $|V| < t$, as the adversary can corrupt atmax t shares.So as $|V'| \leq |V|$, $|V'| < t$, so we fail in the if condition at line number 16, so in this case we won't be updating X with this V'

– Case 2 b):

The other case is when V contain atleast one good share But when we pass this V through filter_share function we will be getting {} empty set. To prove this lets consider the filter_share function, while we are iterating through each share , and let say we are currently at S_i which is a good share, we know that there are few corrupt shares in V because of which now the hash values mismatch happens (assuming hash collision with negligible probability).

So as there is a hash value mismatch we return {} in line number 5 of filter_shares function. So in this case also we won't be updating X with V' as $|V| < t$.

So in this case 2, where we assume V has some bad share, then we are not at all updating X with V.

As only the above 2 cases occurs (and they're exhaustive),we are updating X with all good shares when number of good shares are greater than threshold else X will be empty and this will fail only if there is hash function collision which happens with $negl(\lambda)$ probability, that concludes the proof for the above lemma.

Proof for $Adv^{errx}_{\mathbb{ES}}(\mathbb{A}) \leq negl(\lambda)$ when $\mathbb{ES} = EX[\mathbb{SS}]$ and $\mathbb{SS}$ is fully correct:

We need to show $Pr[\mathbb{ES}.Recover(A, S) \neq UniqueExplanation^H(K, S)] \leq negl(\lambda)$

let $E = Explanations^H(K, S)$

- Case 1: $E = \Phi$, This is trivial case where in $\mathbb{ES}.Recover(K, S)$ will be returning $\perp$, because if it returns let's say (M, R, G) then we'll be having one explanation which contradicts the fact that Explanations are empty.
  So in this case $Pr[\mathbb{ES}.Recover(A, S) \neq UniqueExplanation^H(K, S)] = 0$.

- Case 2: $E \neq \Phi$

  In this case we can show that $Pr[UniqueExplanation^H(K, S) = \perp] \leq negl(\lambda)$. In order to have an explanation the set X shouldn't be Φ for some input $\mathbb{ES}.Recover^H(K, S'), S' \subseteq S$, because if X is empty then we return $\perp$ in Recover function. So S' should have atleast t good shares then only the $X \neq \Phi$, this we have seen during the case 1 proof for claim 4, as $S' \subseteq S$, so S will also be

having atleast t good shares.

So as S contains atleast threshold number of good shares, in $\mathbb{ES}.Recover^H(K, S)$ we get X = all good shares in S by claim 4, and as $|X| \geq t$, X is an authorized subset.So by full correctness assumption $(M, R, X) \leftarrow \mathbb{SS}.Recover(K, X)$. So in line number 20 we'll be getting (M, R, X) and we return (M, R, G) as the output of $\mathbb{ES}.Recover^H(K, S)$ where G is just hash value concatenations of X, and G is set of all good shares of S.

Now we need to prove $UniqueExplanation^H(K, S) = (M, R, G)$ assuming hash function is collision resistant, As we have $(M, R, G) \leftarrow ES.Recover^H(K, S)$, so $(M, R, G) \in Explanations^H(K, S)$. Now we can show that for any $S' \subseteq S$ if $(M', R', g) \leftarrow \mathbb{ES}.Recover^H(K, S')$ then we'll be having $(M = M')$ and $(R = R')$ and $g \subseteq G$.

As we are getting some (M', R', g) as output for the input (K, S') to the Recover algorithm, it means S' must be having not corrupted shares ≥ threshold, and as per our previous discussions or claim 4 the X must be set of all good shares , and as $|X| \geq t$, X is an authorized subset so $(M', R', X) \leftarrow \mathbb{SS}.Recover^H(K, X)$, by full correctness property of $\mathbb{SS}(M', R') = (M, R)$ and next g is just hash value concatenation extension to X. So g is some set of good shares , so it must be $\subseteq$ of G. As G is the set of good shares.

So we now proved that for any $S' \subseteq S$ if $(M', R', g) \leftarrow \mathbb{ES}.Recover^H(K, S')$ then we'll be having $(M = M')$ and $(R = R')$ and $g \subseteq G$, this means that (M, R, G) is the $UniqueExplanation^H(K, S)$. Hence $\mathbb{ES}.Recover^H(K, S) = UniqueExplanation^H(K, S)$ and it fails to happen only when there is collision in hash function values, which happens at $negl(\lambda)$ probability.

So $Pr[\mathbb{ES}.Recover(A, S) \neq UniqueExplanation^H(K, S)] \leq negl(\lambda)$ and $Pr[UniqueExplanation^H(K, S) = \perp] \leq negl(\lambda)$.

In both cases we showed $Pr[\mathbb{ES}.Recover(A, S) \neq UniqueExplanation^H(K, S)] \leq negl(\lambda)$ So $Adv^{errx}_{\mathbb{ES}}(\mathbb{A}) \leq negl(\lambda)$ that completes the proof.

Proof for if $\mathbb{SS}$ is auth-secure then $\mathbb{ES}$ is auth-secure:

For the sake of contradiction let's assume $\mathbb{ES}$ is not auth-secure and $\mathbb{SS}$ is auth-secureThen we must be able to find $(M, R, G) \leftarrow \mathbb{ES}.Recover(A, S_1^{'})$ and $(M', R', G') \leftarrow \mathbb{ES}.Recover(A, S_2^{'})$ with $G \cap G' \neq \Phi$ and $(M', R') \neq (M, R)$ as $\mathbb{ES}$ is not auth-secure. If we look at line number 23 of the $\mathbb{ES}.Recover^H(K, S)$ algorithm , we get the output

(M, R, G) only if there is some $(M, R, D) \leftarrow \mathbb{SS}.Recover^{H}(K, V)$ in line 20.

Now for the $(M, R, G) \leftarrow \mathbb{ES}.Recover(A, S_1^{'})$ and $(M', R', G') \leftarrow \mathbb{ES}.Recover(A, S_2^{'})$ there will be corresponding $(M, R, D) \leftarrow \mathbb{SS}.Recover(A, V)$ and $(M', R', D') \leftarrow \mathbb{SS}.Recover(A, V')$ $D \cap D' \neq \Phi$ as $G \cap G' \neq \Phi$ and G is just some concatenation of the hash values of all shares in D.

We have $(M, R, D) \leftarrow \mathbb{SS}.Recover(A, V)$ and $(M', R', D') \leftarrow \mathbb{SS}.Recover(A, V')$ and $D \cap D' \neq \Phi$, with $(M, R) \neq (M', R')$,we are able to find this sets of shares in polynomial time as $\mathbb{ES}$ is not auth-secure.

Now this contradicts the fact that $\mathbb{SS}$ is auth-secure. Hence by the proof of contradiction if $\mathbb{SS}$ is secure then $\mathbb{ES}$ is also auth-secure.

APPENDIX A

New Errx Definition and Construction

Before presenting the definition and construction we made few assumptions on Adversary $\mathbb{A}$, and we deal here only with threshold schemes, let say the threshold is t.

Adverserial Assumptions:

- It is safe to assume Adversary A can't see all t shares , because then privacy itself will be lost and there's no point in moving forward.
- So he can't infiltrate/corrupt all t shares, and can atmax corrupt $t-1$ shares.
- So if recover algorithm get atleast t shares then we are sure that atleast 1 share is pure and not got corrupted.
- We also assume that Recover algorithm knows the Access Structure or in this case the threshold t, else simple attack is that adversary corrupt few shares and keep threshold as 1 in Access structure field of the shares tag this could make Error recovery process always return $\perp$ because of multiple explanations.

Motivations:

The error recovery algorithm that ADSS use can be termed as explanation seeking, and if there is any unique explanation we are outputting the message else returning $\perp$ symbol. But now we are aiming at a scheme which recovers the message when there is honest majority and else returns $\perp$ symbol.

A.1 New_Errx definition

Game $G_{\mathbb{S}}^{New\_errx}(\mathbb{A})$

$H \leftarrow \Omega$;
$(A, M, R, T) \leftarrow \mathbb{A}^H$
$S \leftarrow \mathbb{S}.Share^H(A, M, R, T)$
$\mathbb{A}$ can choose to corrupt atmax t-1 shares where t is threshold.
V be the final set of shares after Adversary $\mathbb{A}$ corrupts max t-1 shares of S
$S' \leftarrow \mathbb{A}^H(V)$, where $S' \subseteq V$
$k \leftarrow$ good_shares_cnt(S')
if $k \geq t$ **then**
    return $(M, R, G) \neq \mathbb{S}.Recover^H(A, S')$ for some $G \subseteq S'$
end if
return $\perp \neq \mathbb{S}.Recover^H(A, S')$ and $(M, R, G) \neq \mathbb{S}.Recover^H(A, S')$ for some $G \subseteq S'$

The above game represents the robustness aims that we want for Error recovery process. It basically says if there are atleast threshold number of good shares that are not corrupted, then we must recover the actual message, else we can't recover or we may recover only actual message but not other random messages.

Details of Game :

- Adversary $\mathbb{A}$ chooses (A, M, R, T) of it's choice, then share algorithm is ran on this input to generate vector of shares S.
- Now Adversary $\mathbb{A}$ is allowed to corrupt at max $t - 1$ shares, where t is the threshold of the access structure A.
- Now after these few shares that got corrupted by $\mathbb{A}$, we ask adversary to choose any subset of shares and let's call it S'
- let k be the number of good shares in S' that are not corrupted.
- We say Adversary $\mathbb{A}$ wins the game if it forces the recovery to recover something wrong when there is a honest majority (i.e., $k >= t$).
- If there is no honest majority but still if Recover algorithm recovers something other than actual (M, R) then Adversary wins the game.

$$\text{We define } Adv_{\mathbb{S}}^{New\_errx}(\mathbb{A}) = Pr[G_{\mathbb{S}}^{New\_errx}(\mathbb{A})]$$

Next we present a scheme which we call more robust scheme called as $\mathbb{ES} = EX[\mathbb{SS}]$ with $\mathbb{SS}$ is the Adept Secret sharing scheme with full correctness and $\mathbb{SS}$ is Auth secure.

A.2 $\mathbb{ES}$ Construction

procedure $\mathbb{ES}.Share^{H}(\mathcal{A}, M, R, T)$

$S \leftarrow \mathbb{SS}.Share^{H}(\mathcal{A}, M, R, T)$
$n \leftarrow n(Acc(\mathcal{A}))$
$(S_1, \ldots S_n) \leftarrow S$
for $i \in [1 \ldots n]$ **do**
    $S_i^{'} \leftarrow S_i || H(S_1) || H(S_2) || \ldots || H(S_n)$
end for
return $(S_1^{'}, S_2^{'}, \ldots, S_n^{'})$

In the share algorithm we initially run the Share algorithm of $\mathbb{SS}$ to get the shares and then for every share S_i we concatenate the hash value of all the shares $H(S_1)||H(S_2)\ldots||H(S_n)$ and call it as $S_i^{'}$ now this vector $S^{'}$ will be the shares outputted by this share algorithm.

The Recover Algorithm is :

procedure $\mathbb{ES}.Recover^{H}(K, S)$

K is the known information which is Access structure A
$S \leftarrow \{x : x \in S \text{ and } x.access = K\}$
$t \leftarrow |S|$;
for $i \in [1..t]$ **do**
    $(a_i, S_i^{'}) \leftarrow S_i$
end for
for $i \in [1 \ldots t]$ **do**
    $V = \{\}$
    for $j \in [1 \ldots t]$ **do**
        if $S_i^{'}[a_j] = H(S_j[0])$ **then**
            V.add$(S_j[0])$
        end if
    end for
    $(M, R, D) = \mathbb{SS}.Recover^{H}(K, V)$
    if $(M, R, D) \neq \perp$ **then**
        $G \leftarrow concat\_hash\_values(D)$
        return (M, R, G)
    end if
end for
return $\perp$

In the recover algorithm at the start we'll filtering the shares that are having

same access structure field that is known to the recoverer. Later we extract the actual share numbers fields of each share in a_i.Now we run for loop over size t , in each iteration check in the context of the current share S_i' which all the shares are correct by comparing the Hash value that this share has maintained. Now we pass this set V to the Recover algorithm of $\mathbb{SS}$ and if we get a message (M, R, S) as output then we output it. If we are not able to get any message as output then we return $\perp$ symbol.

Theorem 6. *Let* $\mathbb{ES} = EX[\mathbb{SS}]$ *with* $\mathbb{SS}$ *be any enriched ADSS scheme with full correctness, then if* $\mathbb{SS}$ *is auth-secure then* $\mathbb{ES}$ *is auth-secure and* $Adv_{\mathbb{ES}}^{New\_errx}(\mathbb{A})$ *is* $negl(n)$

Proof for $\mathbb{ES} = EX[\mathbb{ES}]$ is auth-secure if $\mathbb{SS}$ is auth-secure:

- For the sake of contradiction let's assume $\mathbb{ES}$ is not auth-secure and $\mathbb{SS}$ is auth-secure.
- Then we must be able to find $(M, R, G) \leftarrow \mathbb{ES}.Recover(A, S_1^{'})$ and $(M', R', G') \leftarrow \mathbb{ES}.Recover(A, S_2^{'})$ with $G \cap G' \neq \Phi$ and $(M', R') \neq (M, R)$ as $\mathbb{ES}$ is not auth-secure.
- If we look at line number 15 of the $\mathbb{ES}.Recover^H(K, S)$ algorithm , we get the output (M, R, G) only if there is some $(M, R, D) \leftarrow \mathbb{SS}.Recover^H(K, V)$ in line 14.
- Now for the $(M, R, G) \leftarrow \mathbb{ES}.Recover(A, S_1^{'})$ and $(M', R', G') \leftarrow \mathbb{ES}.Recover(A, S_2^{'})$ there will be corresponding $(M, R, D) \leftarrow \mathbb{SS}.Recover(A, V)$ and $(M', R', D') \leftarrow \mathbb{SS}.Recover(A, V')$
- $D \cap D' \neq \Phi$ as $G \cap G' \neq \Phi$ and G is just some concatenation of the hash values of all shares in D.
- We have $(M, R, D) \leftarrow \mathbb{SS}.Recover(A, V)$ and $(M', R', D') \leftarrow \mathbb{SS}.Recover(A, V')$ and $D \cap D' \neq \Phi$, with $(M, R) \neq (M', R')$,we are able to find this sets of shares in polynomial time as $\mathbb{ES}$ is not auth-secure.
- Now this contradicts the fact that $\mathbb{SS}$ is auth-secure.
- Hence by the proof of contradiction if $\mathbb{SS}$ is secure then $\mathbb{ES}$ is also auth-secure.

Lemma 7. *If* $\mathbb{SS}$ *is fully correct ADSS scheme, then* $\mathbb{ES} \leftarrow EX[\mathbb{SS}]$ *is also fully correct.*

Proof: For the fully correctness we assume that the shares we are dealing with are not corrupted, then in that case we'll be having same (M, R) output for both $\mathbb{ES}.Recover(A, S)$ and $\mathbb{SS}.Recover(A, S)$, this is because all shares are good , so the set $V = S$, and we'll be calling $\mathbb{SS}.Recover(A, S)$ in line number 14, to get the message (M, R) all the time. So as (M, R) outputs are same for both $\mathbb{ES}$ and $\mathbb{SS}$ schemes , the fully correctness properties will be equivalent for both , i.e., if $\mathbb{SS}$ is fully correct then $\mathbb{ES}$ is also fully correct.

Proof for $Adv_{\mathbb{ES}}^{New\_errx}(\mathbb{A}) \leq negl(n)$ when $\mathbb{ES} = EX[\mathbb{SS}]$ and $\mathbb{SS}$ is auth secure:

In the *New_errx* game, let's consider two cases $k < t$ and $k \geq t$

Case 1: $k < t$

- In this case the number of good shares that are sent to recover algorithm are less than threshold number
- Now we need to prove if $(M', R', G) \leftarrow \mathbb{ES}.Recover(A, V)$ with $G \subseteq V$ then $(M', R') \neq (M, R)$ happens with negligible probability.
- We can show that $G \cap S$ is not empty, because if $G \cap S$ is empty then all members of G are just corrupted shares and we also have $|G| < t$ as adversary can't corrupt t or more shares. So as $|G| < t$ then recover algorithm will return $\perp$ as number of shares needed for construction are less than threshold, hence we prove that $G \cap S$ is not empty.
- We can easily see that $\mathbb{ES}.Recover(A, S) = (M, R, S)$ as all are good shares and we proved earlier that $\mathbb{ES}$ is the full correct , so we should be getting (M, R, S) as output.
- Now if $(M', R', G) \leftarrow \mathbb{ES}.Recover(A, V)$ and $(M, R, S) \leftarrow \mathbb{ES}.Recover(A, S)$ and $G \cap S \neq \Phi$.
- If $(M', R') \neq (M, R)$ it means we broke the authenticity of $\mathbb{SS}$ which happens at negligible probability as $\mathbb{ES}$ is Auth secure that we proved previously.
- Hence $\mathbb{ES}.Recover(A, V)$ returns (M', R', G) with $G \subseteq V$ and $(M', R') \neq (M, R)$ with negligible probability.

Case 2: $k \geq t$

- In this case the number of good shares that Adversary send to the recovery algorithm in line number 9 are greater than threshold.

- Here we need to prove $(M, R, G) \neq \mathbb{ES}.Recover^H(A, S')$ happens with negligible probability.
- Case 2a) :
  - Let's calculate $\Pr[\mathbb{ES}.Recover^H(A, S') = \perp]$ we can prove that this Probability is negligible , because there is a honest majority , and let's say we are considering some good share in the for loop in line number 7 of $\mathbb{SS}.Recover$, in the context of this good share the set V will be filled with all the good shares which are not corrupted.
  - the set V can have corrupted shares only if there is a hash collision for H which again happens with negligible probability, so we can assume V is set of all good shares.
  - now $|V| \geq t$ as number of good shares are greater than threshold, so $V \in \mathcal{A}$ i.e., V is authorized subset.
  - So by the full correctness assumption of the scheme $\mathbb{SS}$ we'll get the output (M, R, G)
  - Hence we return $\perp$ with negligible probability.
- Case 2b):
  - Let's calculate $\Pr[(M', R', G) \leftarrow \mathbb{ES}.Recover^H(A, S')$ and $(M', R') \neq (M, R)]$, we can prove that this probability will be negligible.
  - We claim that G must contain atleast 1 good share , otherwise $|G| < t$ as number of corrupt shares are less than t. So $G \cap S \neq \Phi$.
  - We already know $(M, R, S) \leftarrow \mathbb{ES}.Recover(A, S)$, because $\mathbb{ES}$ is full correct, and S is an authorized subset of all good shares.
  - So now we have $(M', R', G) \leftarrow \mathbb{ES}.Recover^H(A, S')$ and $(M, R, S) \leftarrow \mathbb{ES}.Recover(A, S)$ and $G \cap S \neq \Phi$.
  - $(M', R') \neq (M, R)$ happens only with negligible probability as $\mathbb{ES}$ is Auth-secure.
  - Hence $\Pr[(M', R', G) \leftarrow \mathbb{ES}.Recover^H(A, S')$ and $(M', R') \neq (M, R)]$ is negligible.
- $\Pr[(M, R, G) \neq \mathbb{ES}.Recover^H(A, S')] = \Pr[\mathbb{ES}.Recover^H(A, S') = \perp] + \Pr[(M', R', G) \leftarrow \mathbb{ES}.Recover^H(A, S')]$ which is negligible.

  So both the cases happen with negligible probability and Probability of the game returning true is negligible, therefore $\mathbb{ES} = EX[\mathbb{SS}]$ is New_errx secure when $\mathbb{SS}$ is fully correct and Auth-secure.

REFERENCES

Bellare, **Dail**, and **Rogaway** (2020). Reimagining secret sharing: Creating a safer and more versatile primitive by adding authenticity, correcting errors, and reducing randomness requirements. URL https://eprint.iacr.org/2020/800.pdf.

Shamir, A. (1979). How to share a secret. URL https://web.mit.edu/6.857/OldStuff/Fall03/ref/Shamir-HowToShareASecret.pdf.