On the Power of Computational Secret Sharing

V. Vinod[1*], Arvind Narayanan[2], K. Srinathan[2**], C. Pandu Rangan[2***], and Kwangjo Kim[3]

[1] Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.

[2] Department of Computer Science and Engineering, Indian Institute of Technology, Madras, Chennai - 600036, India.

[3] IRIS (International Center for Information Security), ICU (Information and Communications University) 58-4, Hwaam-Dong, Yusong-gu, Taejon 305-732, South Korea.

Email: `vinodv@mit.edu, arvindn@meenakshi.iitm.ernet.in, ksrinath@cs.iitm.ernet.in, rangan@iitm.ernet.in, kkj@icu.ac.kr`

Abstract. Secret sharing is a very important primitive in cryptography and distributed computing. In this work, we consider computational secret sharing (CSS) which provably allows a smaller share size (and hence greater efficiency) than its information-theoretic counterparts. Extant CSS schemes result in succinct share-size and are in a few cases, like threshold access structures, optimal. However, in general, they are not *efficient* (share-size not polynomial in the number of players n), since they either assume efficient perfect schemes for the given access structure (as in [10]) or make use of exponential (in n) amount of public information (like in [5]). In this paper, our goal is to explore other classes of access structures that admit of efficient CSS, without making any other assumptions. We construct efficient CSS schemes for every access structure in monotone P. As of now, most of the efficient information-theoretic schemes known are for access structures in algebraic NC^2. Monotone P and algebraic NC^2 are not comparable in the sense one does not include other. Thus our work leads to secret sharing schemes for a new class of access structures. In the second part of the paper, we introduce the notion of secret sharing with a semi-trusted third party, and prove that in this relaxed model efficient CSS schemes exist for a wider class of access structures, namely monotone NP.

Keywords: Secret sharing, computationally bounded players, access structures, monotone P.

1 Introduction

Secret sharing schemes protect the secrecy and integrity of information by distributing the information over different locations (not necessarily geographical).

---

* Work done when the author was at the Indian Institute of Technology, Madras.

** Financial support from Infosys Technologies Limited, India is acknowledged.

*** Financial support from the Ministry of Information Technology, Government of Korea is acknowledged.

This forces the *adversary* to attack multiple locations in order to learn or destroy the information. In a secret sharing protocol, the *dealer* shares his secret among n players. In the so called *threshold* model, the sharing is done so that subsets of $t+1$ (or more) players can later correctly reconstruct the secret, while subsets of t (or fewer) players cannot reconstruct it. This notion can be generalized by specifying a family of authorized subsets of the n players, called *access structures*. The dealer shares the secret in such a way that only authorized subsets of players can reconstruct the secret while the players in non-authorized subsets cannot.

1.1 Why Computational Secret Sharing ?

CSS schemes allow us to achieve a better *information rate* than is possible with information theoretic schemes. The information rate of a secret sharing scheme is the maximum length of a (player's) share per unit size of the secret. This measure is of interest when the size of the message to be shared is large. For instance, it is possible ([10]) to t-share a secret, in the threshold model, with the share of each player being only $\frac{1}{t}$ as long as the secret (which is clearly optimal). Indeed, Beguin and Cresti [1] have shown that it is possible to achieve the optimal information rate for *any* access structure provided that a fixed length secret can be shared on the same access structure. The above schemes build on the idea of *information dispersal algorithms* [13]. CSS schemes which make use of a *bulletin board* on which an arbitrary amount of information can be published have also been proposed ([5]).

Clearly, a secret sharing scheme can be called *efficient* only if the information rate is polynomially bounded as a function of the number of players n. Furthermore, we posit that the amount of information published on the bulletin board be polynomial in n. Note that the CSS of [1] is not applicable when there is no known efficient secret sharing scheme for the corresponding access structure. Similarly, the CSS scheme of [5] becomes inefficient when the size of the access structure is not polynomial in n. Our interest in studying CSS is due to the fact that there may exist access structures that have efficient CSS schemes but do not permit any other (information theoretic) efficient secret sharing scheme.

Most of the results in extant literature on secret sharing schemes deal with information-theoretic secret sharing. We know that efficient information-theoretic perfect linear secret sharing schemes exist only if the access structure is in algebraic $NC^2 \cap mono$, the class of monotone languages which can be computed by algebraic circuits of logarithmic size and log-squared depth. Non-linear schemes appear to be more powerful [2]. However, there remains a large gap between the known upper bounds and lower bounds in the case of information theoretic secret sharing.

1.2 Boolean formulas and Boolean circuits

The contribution of the first part of this paper is the construction of efficient CSS schemes for all access structures which can be computed by monotone Boolean circuits of polynomial size. This complexity class is known as mP, for monotone P. Monotone Boolean circuits differ from monotone Boolean formulas in that in the former, the output of a node can serve as the input of more than one node. We say that gates in a monotone Boolean circuit can have a *fanout* of more than one. Another way of understanding the difference is that a monotone Boolean circuit is a *directed acyclic graph* where each node represents an AND gate or an OR gate whereas a monotone Boolean formula is a *tree* with the same property.

This makes monotone Boolean circuits much more powerful than formulas. In fact, it was believed that monotone Boolean circuits could simulate any (deterministic) Turing machine accepting a monotone set (thus making mP equivalent to $P \cap mono$, the class of monotone languages in P), but Razborov ([14]) proved that this is not the case.[1]

Benaloh and Leichter [3], in their landmark result on the existence of linear perfect secret sharing schemes for any monotone access structure (as first defined in [8]), showed how to combine secret sharing schemes across AND and OR gates (in other words, how to realize secret sharing schemes for the union and intersection of two access structures) and recursively applied this result to the Boolean formula computing the access structure. Our construction is similar in spirit. We represent the Boolean circuit computing the access structure as a graph whose nodes are AND, OR, or $FANOUT$ gates. (The $FANOUT$ gate takes a single input and produces multiple copies of the input. We do this in order that the $FANOUT$ is the only gate with more than one output.) With each edge of this graph, we associate a (virtual) secret, which we call the share of that edge. The shares of the input wires of the circuit form the shares of the players. The sharing scheme has the property that a subset S of players can compute the share of some edge E if the wire corresponding to E evaluates to 1 when the circuit is given (the encoding of) S as input. We show how to associate shares with edges in such a way that the above property is carried across AND, OR and $FANOUT$ gates.

Our techniques are similar in spirit to Yao's landmark garbled circuit construction ([15]), but very different in application since in the case of secret sharing, non-interactivity is essential. Thus our result does not follow from Yao's since secret sharing protocols cannot be expressed as special case(s) of interactive secure function evaluation protocols.

1.3 Semi-trusted third party

Our second result deals with secret sharing using a semi-trusted third party. The use of this construct is to introduce a limited amount of interactivity into the protocol, and thus increase its power. Just as the relaxation of the security requirement from information theoretic to computational security allows us to give protocols for a broader class of access structures, the relaxation of the non-interactivity requirement results in a further broadening. We prove that, using

[1] Razborov showed superpolynomial lower bounds for the monotone circuit complexity of the *matching* function.

a semi-trusted third party, efficient CSS schemes exist for any class of access structures in monotone NP (denoted mNP). This is the class of languages accepted by monotone non-deterministic Turing machines in polynomial time, which also turns out to be the class of monotone languages in NP. Clearly, this includes all access structures that could possibly be interesting in practice.

The notion of a semi-trusted third party has been made use of in protocols for *fair exchange* ([6]). It allows two parties to exchange a secret in such a way that neither party can gain an unfair advantage by aborting the protocol at any point. To the best of our knowledge, however, secret sharing with a semi-trusted third party has not been considered.

A semi-trusted third party may try to deviate from the protocol, but it cannot collude with any of the players. It cannot be trusted with any of the private information of the other players. Therefore, in the case of secret sharing, we have the restriction that the semi-trusted third party should neither gain knowledge of the secret nor be able to identify the access set of players that tries to determine the secret. We make these notions formal in the next section.

Semi-trusted third parties are worthy of study because, unlike trusted third parties, they are readily realizable in practice. Indeed, [6] have suggested that in networks such as the internet, a *random* player can be chosen as a semi-trusted third party. In such a scenario, the third party is both geographically and logically separated from the players, and thus the possibility of both the third party and some of the other players coming under the control of a common adversary is remote. Another practical possibility is for a bank to play the role of a semi-trusted third party.

2 Preliminaries and Definitions

To begin with, we formally define the concept of Computational Secret Sharing for general access structures. A Secret Sharing scheme is a protocol between the set of players $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ and a dealer D, where we assume $D \notin \mathcal{P}$. An access structure $\mathcal{A} \subseteq 2^{\mathcal{P}}$ consists of sets of players qualified to recover the secret. It is natural to consider only monotone access structures $\mathcal{A}$, that is, if $A \in \mathcal{A}$ and $A \subseteq A' \subseteq \mathcal{P}$, then $A' \in \mathcal{A}$. The set $\bar{\mathcal{A}} = 2^{\mathcal{P}} - \mathcal{A}$ is called the *adversary structure*. A set of players $A \in \mathcal{A}$ is called an *access set* or a *qualified subset*. A set of players $A \notin \mathcal{A}$ is called an *adversary set* or a *non-qualified subset*. We associate a class of access structures $\{\mathcal{A}_n\}$ with a language

$$L_{\mathcal{A}} = \{x = x_1 x_2 \ldots x_n : x_i \in \{0, 1\}, \{P_i | x_i = 1\} \in \mathcal{A}_n\}$$

We make statements such as "the access structure $\mathcal{A}$ is in monotone P", when we actually mean to say that $L_{\mathcal{A}} \in$ monotone P.

The set of all possible secrets is called the *secret domain* (denoted by $\mathbb{S}$) and the set of all possible shares is called the *share domain* (denoted by $\mathbb{S}'$). Now, we formally define a computational secret sharing scheme.

Definition 1 (Computational Secret Sharing). *A computational secret sharing scheme is a protocol π between D and $\mathcal{P}$ to share a secret $S \in \mathbb{S}$, respective to an access structure $\mathcal{A}$ such that*

- *The dealer D transmits a share $S_i \in \mathbb{S}'$ to the player P_i, for $i = 1, 2, \ldots, n$. D retires from the protocol immediately afterwards.*
- *There is a polynomial-time algorithm π_{REC} such that $\pi_{REC}(S_{i_1}, S_{i_2}, \ldots, S_{i_m}) = S$ with probability 1 if $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\} \in \mathcal{A}$.*
- *For any set of players $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\} \notin \mathcal{A}$ and any (possibly randomized) polynomial-time algorithm π_{ADV}, $Prob[\pi_{ADV}(S_{i_1}, S_{i_2}, \ldots, S_{i_m}) = S] \leq \frac{1}{|\mathbb{S}|^c}$ for all constants c and suitably chosen $|\mathbb{S}|$.*

In our secret-sharing schemes, the domains of the secret and of the shares are the same, and this common domain is a finite field, which we denote by $\mathcal{F}$.

Definition 2 (Secret sharing using a semi-trusted third party). *A computational secret sharing scheme using a semi-trusted third party is a pair of protocols σ and ρ between a dealer D, the set of players $\mathcal{P}$ and a third party T, to share a secret $S \in \mathbb{S}$, respective to an access structure $\mathcal{A}$ such that*

- *In the sharing protocol σ, the dealer D transmits a share $S_i \in \mathbb{S}'$ to the player P_i, for $i = 1, 2, \ldots, n$, and a share S_0 to T. D retires from the protocol immediately afterwards.*
- *The reconstruction protocol ρ is an interactive protocol between T and some subset $A \subset \mathcal{P}$, represented by the virtual player P, at the end of which:*
  - *T should not obtain any information about S or about A.*
  - *If $A \in \mathcal{A}$ then P should be able to compute the secret S with certainty in polynomial time.*
  - *If $A \notin \mathcal{A}$, then the probability that P can compute S in polynomial time should be negligible.*

Monotone Boolean circuit. A monotone Boolean circuit is a Boolean circuit consisting of AND, OR and $FANOUT$ *gates* connected by *wires*. Both AND and OR gates have two inputs and one output; $FANOUT$ gates have one input and two outputs. AND and OR gates perform Boolean multiplication and addition respectively on their inputs, while the $FANOUT$ gates propagate their input to both outputs. The circuit has n input wires (which are not the output of any gate) and one output wire (which is not the input of any gate).

There are two values associated with each wire W of the circuit - the Boolean value of W obtained by the evaluation of the circuit on some input assignment and the share value associated with W during the sharing and reconstruction process. The Boolean value of W corresponding to an input assignment $x_1, x_2, \ldots, x_n$ is denoted by $Eval(W, A)$ where $(x_1, x_2, \ldots x_n)$ is the encoding of A. We abuse notation and denote $Eval(W, A)$ by $Eval(W)$ when it is clear which set of players A we are referring to. Given a wire W in the circuit, we denote by $V(W)$ the share-value associated with W.

Definition 3 (Nondeterministic Boolean circuit). *A nondeterministic circuit for a Boolean function $f(x_1, x_2, \ldots x_n)$ is a circuit C with standard inputs $x_1, x_2, \ldots x_n$ and auxiliary inputs $y_1, y_2, \ldots y_m$, where $m = poly(n)$, such that if $f(x_1, x_2, \ldots x_n) = 1$, then there is a assignment for the inputs y such that $C(x_1, x_2, \ldots x_n, y_1, y_2, \ldots y_m) = 1$ and if $f(x_1, x_2, \ldots x_n) = 0$, then there is no such assignment.*

We model such a circuit as a directed acyclic graph whose nodes are AND, OR, NOT, or $FANOUT$ gates.

Definition 4 (Monotone nondeterministic Boolean circuit). *A monotone nondeterministic Boolean circuit is a nondeterministic Boolean circuit in which a gate that (transitively) depends on a standard input*[2] *cannot be a NOT gate.*

Given an access structure $\mathcal{A}$ in mNP, we associate with it the monotone nondeterministic Boolean circuit for the characteristic function of the language $\mathcal{L}_{\mathcal{A}}$.

Lemma 1 ([7]). *A language $\mathcal{L}$ is in mNP if and only if the monotone nondeterministic Boolean circuit computing $\mathcal{L}$ has polynomial size.*

Definition 5 (Oblivious transfer). *An oblivious transfer (or OT) protocol is a protocol between a sender S and a receiver R in which*

- *S's input is (s_1, s_2) which are elements of the secret domain $\mathbb{S}$.*
- *R's input is an index $\alpha \in \{0, 1\}$.*
- *At the end of the protocol R must obtain s_α but should not get any information about $s_{1-\alpha}$.*
- *S should not obtain any information about α.*

The definition above refers to "1-out-of-2 OT", or OT_1^2. There are more general notions of oblivious transfer, but we will not require them.

3 Our Computational Secret Sharing Scheme

We assume that the players are provided a monotone Boolean circuit C that accepts the access structure $\mathcal{A}$ (the circuit can be individually given to all the players or published on a bulletin board). We consider the circuit as composed of AND, OR and $FANOUT$ gates. Each wire of the circuit will be associated with a value during the sharing and reconstruction phases. In the sharing phase, the problem is to compute the values corresponding to the input wires (the shares of the corresponding players) from the value of the output wire (the secret). We perform the reverse of this during reconstruction. Our computational secret sharing scheme is as follows.

[2] In other words, there is a directed path from a standard input to that gate.

Let $ENC_K : \mathcal{F} \to \mathcal{F}$ be a family of trapdoor one-way functions[3] on $\mathcal{F}$ with the index K varying over $\mathcal{F}$, and $DEC_K : \mathcal{F} \to \mathcal{F}$ the corresponding inverses.

Algorithm Share

1. Let W be the output wire. Assign $V(W) = s$, where s is the secret.
2. Choose a gate G whose output wire has been assigned a value.[4]
   - G is an AND gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Pick a random x in $\mathcal{F}$. Assign: $V(W_1) = x \oplus V(W)$ and $V(W_2) = x$.
   - G is an OR gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Assign: $V(W_1) = V(W)$ and $V(W_2) = V(W)$.
   - G is a $FANOUT$ gate : Let W_1, W_2 be the outputs of G and W be the input. Pick a random key K from $\mathcal{F}$. Publish: $ENC_K(V(W_i))$ for $i = 1, 2$. Assign: $V(W) = K$.
3. Repeat step 2 until all gates are considered. The values at each input wire form the shares of the corresponding players.

Algorithm Reconstruct

1. Let W_P be the input wire corresponding to player P. Assign $V(W_P)$ to the share of player P. Choose a gate G whose input wires have been assigned values.
   - G is an AND gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Assign $V(W) = V(W_1) \oplus V(W_2)$.
   - G is an OR gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Choose the input wire $W_i (i = 1, 2)$ such that $Eval(W_i) = 1$. Assign $V(W) = V(W_i)$.
   - G is a $FANOUT$ gate : Let W be the input wire of G and W_1, W_2 be the outputs. By applying DEC, compute $V(W_i)$ from $V(W)$ and $ENC_{V(W)}(V(W_i))$.
2. Repeat step 1 until $V(W_O)$ for the output wire W_O is constructed. $V(W_O)$ is the secret.

3.1 Correctness and security

Correctness. It is easy to show that any access set A can recover the secret with probability 1. We prove the stronger statement that A can recover $V(W)$ for any wire W with $Eval(W, A) = 1$. We show this by induction on the depth of the W: it is clearly true for the input wires. If W is any other wire, it must be the output of an AND, OR or $FANOUT$ gate G. In each of these cases algorithm reconstruct shows how to obtain $V(W)$ from the V-values at those input wires of G for which $Eval$ is 1 (the V-values at the input wires, which have a smaller depth than W, are assumed to be known by the induction hypothesis). $\square$

[3] We have used trapdoor functions for clarity of presentation, but the protocol will work with minor modifications even when ENC is a one-way function.

[4] If G is a $FANOUT$ gate, both its outputs should have been assigned a value.

Theorem 1. *The above computational secret sharing scheme is secure, i.e, for any $A \notin \mathcal{A}$, A cannot recover the secret.*

Proof. We have shown that the access set A can recover $V(W)$ for any wire W with $Eval(W, A) = 1$. The converse of this assertion is not true: A may be obtain $V(W)$ even when $Eval(W, A) = 0$. To see this consider an OR gate whose inputs W_1 and W_2 evaluate to 1 and 0 respectively. Then A can obtain $V(W_2) = V(W_1)$. To get around this difficulty, we introduce the concept of a *fanout-free region*. It will turn out that the converse statement indeed holds on those wires that connect fanout-free regions.

A subcircuit C' of a circuit C is a connected subgraph of C induced by some of the nodes (gates). A fanout free region (FFR) of C is a maximal subcircuit of C having no $FANOUT$ gates. Note that C can be considered to be a *directed acyclic graph* of fanout free regions connected by $FANOUT$ gates. We call this the *fanout graph* of C. Also note that any FFR is a tree of AND and OR gates.

With every FFR F, we associate a virtual adversary A_F. A_F is given as input the circuit F, the share values of those input wires W of F for which $Eval(W) = 1$, and nothing else. Our goal is to prove that the players' view of the FFR F is indistinguishable from the view of A_F on F.

Let $\pi(W)$ be the property that if $Eval(W)$ is 0 then $V(W)$ is computationally indistinguishable from the uniform distribution on $\mathcal{F}$.

We begin with a restatement of Benaloh and Leichter's result [3]:

Lemma 2. *For the adversary A_F, π holds on the output wire of F.*

The next lemma states that if the input share value of a $FANOUT$ gate is not known, then the public value associated with that gate gives no new information about any of its output share values. In particular this means that the property π is carried across a $FANOUT$ gate.

Lemma 3. *Let W be the input wire and W' an output wire of a $FANOUT$ gate. If $V(W)$ is computationally indistinguishable from the uniform distribution on $\mathcal{F}$, then so is $V(W')$.*

Proof: Since $DEC_K(.)$ is a family of pseudo-random permutations (from $\mathcal{F}$ to $\mathcal{F}$, indexed by the key K), uniform distribution on the key space, given the ciphertext, implies that the distribution on the plaintext space is computationally indistinguishable from uniform distribution. If the distribution of $V(W')$ is computationally distinguishable from the uniform distribution, it means that $V(W)$ is computationally distinguishable from uniform distribution, which is by assumption, false. $\square$

The next lemma formalizes the notion that a $FANOUT$ gate does not "leak any information" in the reverse direction.

Lemma 4. *Let W be the input wire and W' be an output wire of a $FANOUT$ gate. Then the distributions $V(W)$ and $V(W)|V(W')$ are computationally indistinguishable.*

We observe that the sharing algorithm fixes $V(W)$ randomly and independently of $V(W')$. Therefore, if the lemma is false it would mean that the knowledge of an arbitrary $(plaintext, ciphertext)$ pair gives information about the key, which contradicts the assumption that ENC is secure. □

A simple generalization of the above lemma to any pair of wires with the property that any path connecting them must pass through a $FANOUT$ gate is:

Lemma 5. *In the fanout graph of C, let F be an FFR of depth d and F' an FFR of depth d', $d' > d$. Let W and W' be wires in F and F' respectively. Then $V(W)$ and $V(W)|V(W')$ are computationally indistinguishable.* □

The above lemma allows us to apply induction on the depth on the fanout graph, at every step ignoring all FFRs at a greater depth than the current FFR.

Lemma 6. *Let F be an FFR at depth d. Assume that π holds on the outputs of all FFRs of depth $< d$. Then π holds on the output of F.*

Proof. By lemma 5, we can ignore the effect of all FFRs of depth $> d$. By assumption, π holds on all wires that feed any of the inputs of F. Applying lemma 3 to each $FANOUT$ gate feeding F, we find that the players' view of the inputs of F is identical to that of A_F. Therefore by lemma 2, π holds on the output of F. □

The rest of the proof is straightforward. By applying induction on the depth d of the FFRs, we find that π holds on the output of every FFR. In particular, π holds on the output wire of the circuit. □

3.2 Efficiency

Theorem 2. *The above scheme is efficient for all access structures $\mathcal{A} \in mP$.*

Proof. The total number of shares given to the players is $O(n)$ since each player gets exactly one share, corresponding to one of the input wires in the circuit. The number of published share values is twice the number of $FANOUT$ gates, which is polynomial in n when the circuit is poly-size. Therefore, for all access structures $\mathcal{A}$ having a polynomial-size circuit (i.e, $\mathcal{A} \in mP$), this scheme is efficient. □

The scheme is also computationally efficient for all $\mathcal{A} \in mP$ since the computational effort required by D is equivalent to that of evaluating the circuit on some input assignment and performing a polynomial number of encryptions. Reconstruction of the secret can be naturally parallelized and the parallel time complexity of reconstructing the secret by an access set $A \in \mathcal{A}$ is proportional to the *depth* of the circuit.

4 Secret sharing with semi-trusted third party

Our goal is to explore the limits on the access structures for which we can give secret sharing schemes by relaxing the requirements. Thus, even though secret sharing as such is a non-interactive protocol, we wish to make it more powerful by allowing a limited amount of interaction. We do this by introducing a third party T who is allowed to interact with the players. However, at the end of the protocol T should be no wiser about the inputs of the dealer and the players than before the beginning of the protocol.

The algorithms for sharing and reconstruction are similar to the first protocol. The main difference is that the circuit consists of NOT gates also. Therefore, we need to associate *two* share values with each wire: one corresponding to the evaluation of the wire being 1 and the other corresponding to the evaluation of the wire being 0. (We denote these by $V(W,1)$ and $V(W,0)$ respectively, and call them the *1-share* and the *0-share* respectively of W.) Propagating the values across AND, OR and $FANOUT$ gates is done as in the previous protocol. In the case of NOT gates, the share value of the input wire with evaluation 0 is related to the share value of the output wire with evaluation 1, and vice versa.

Role of the third party. In monotone circuits, $Eval(W,A) \geq Eval(W,B)$ whenever $A \supset B$. Therefore, the dealer need not worry about a set of players obtaining some share values by evaluating the circuit (i.e, invoking the reconstruction algorithm) with some input x_i set to 0 even though is possible to evaluate the circuit with $x_i = 1$. In the case of a general circuit (which includes NOT gates), however, this is not true, and therefore it is possible that the players might obtain both the 0-share and the 1-share of some wire. The role of the third party is to ensure that this cannot happen. It is enough to ensure that the players cannot get both the 0-share and the 1-share of any auxiliary input wire of the circuit. This is done by executing an *Oblivious Transfer* protocol [12] for each auxiliary input wire of the circuit.

Let $\mathcal{A}$ be an access structure in mNP. By lemma 1, there exists a monotone nondeterministic Boolean circuit C of polynomial size that computes $\mathcal{A}$. Using this circuit we will construct a CSS scheme for $\mathcal{A}$.

4.1 Protocol

Sharing The sharing algorithm is essentially the sharing algorithm of the previous section invoked twice, once for the 0-shares and once for the 1-shares.

1. Let W be the output wire of C. Assign $V(W,1) = s$, where s is the secret.
2. Choose a gate G whose output wire has been assigned a 1-share.
   - G is an AND gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Pick a random x in $\mathcal{F}$. Assign: $V(W_1,1) = x \oplus V(W,1)$ and $V(W_2,1) = x$.
   - G is an OR gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Assign: $V(W_1,1) = V(W,1)$ and $V(W_2,1) = V(W,1)$.
   - G is a NOT gate: Assign $V(W',0) = V(W,1)$ where W is the output wire and W' is the input wire of G.
   - G is a $FANOUT$ gate : Choose a random key K from $\mathcal{F}$. Let W_1, W_2 be the outputs of G and W be the input. Publish: $ENC_K(V(W_i,1))$ for $i = 1,2$. Assign: $V(W,1) = K$.

Choose a gate G whose output wire has been assigned a 0-share.
   - G is an AND gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Assign: $V(W_1, 0) = V(W, 0)$ and $V(W_2, 0) = V(W, 0)$.
   - G is an OR gate : Let W be the output wire of G and W_1 and W_2 be the input wires. Pick a random x in $\mathcal{F}$. Assign: $V(W_0, 0) = x \oplus V(W, 0)$ and $V(W_2, 0) = x$.
   - G is a NOT gate: Assign $V(W', 1) = V(W, 0)$ where W is the output wire and W' is the input wire of G.
   - G is a $FANOUT$ gate : Choose a random key K from $\mathcal{F}$. Let W_1, W_2 be the outputs of G and W be the input. Publish: $ENC_K(V(W_i, 0))$ for $i = 1, 2$. Assign: $V(W, 0) = K$.
3. Repeat steps 2 and 3 until all gates are considered.
4. For every (W, b) that has not been assigned a share, assign a random value to $V(W, b)$.
5. The 1-shares of the input wires form the shares of the corresponding players. The 0-shares of the input wires are published.
6. For every auxiliary input wire W, $\{(V(W, 0), V(W, 1)\}$ is sent to the third party.

Reconstruction The reconstruction consists of two stages: in the first stage the players interact with the third party; in the second the players perform some local computations to recover the secret (if they form an access set).

Stage 1. The players cannot interact individually with the third party through separate channels, because of the requirement that the third party should not gain any information about the set of players involved in the reconstruction procedure. Therefore we consider all the players as constituting one virtual player P.

Let A be the access set of players participating in the reconstruction algorithm. For each auxiliary input wire W:

- T and P execute a OT_1^2 protocol with $V(W, 0)$ and $V(W, 1)$ as T's secrets. Since A is a qualified subset, there exists an assignment of values to the auxiliary inputs y such that the circuit evaluates to 1. For each wire W, P chooses the corresponding value of y as its index.

Stage 2. The stage 2 is similar to the reconstruction phase of the first protocol. The goal is to compute $V(W, Eval(W, A))$ for each wire W. For the input wires these values are already known from the share values and the public information.

1. Choose a gate G whose input wires have been assigned shares.
   - G is an AND gate with output 1: Let W be the output wire of G and W_1 and W_2 be the input wires. Assign $V(W, 1) = V(W_1, 1) \oplus V(W_2, 1)$.
   - G is an OR gate with output 1: Let W be the output wire of G and W_1 and W_2 be the input wires. Choose the input wire $W_i (i = 1, 2)$ such that $Eval(W_i) = 1$. Assign $V(W, 1) = V(W_i, 1)$.

- G is an AND gate with output 0: Let W be the output wire of G and W_1 and W_2 be the input wires. Choose the input wire $W_i (i = 1, 2)$ such that $Eval(W_i) = 0$. Assign $V(W, 0) = V(W_i, 0)$.
- G is an OR gate with output 0: Let W be the output wire of G and W_1 and W_2 be the input wires. Assign $V(W, 0) = V(W_1, 0) \oplus V(W_2, 0)$.
- G is a NOT gate: Let Let W be the output wire of G and W' the input wire. Assign $V(W, b) = V(W', 1 - b)$ where $b = Eval(W)$.
- G is a $FANOUT$ gate : Let W be the input wire of G and W_1, W_2 be the outputs. Apply DEC to compute $V(W_i, b)$ from $V(W, b)$ and $ENC_{V(W,b)}(V(W_i, b))$, where $b = Eval(W)$.

2. Repeat step 1 until $V(W_O, 1)$ for the output wire W_O is constructed. $V(W_O, 1)$ is the secret.

4.2 Correctness and security

To prove the correctness we first note if A is a qualified subset then the circuit evaluates to 1 on the input $(x_1, x_2, \ldots x_n, y_1, y_2, \ldots y_n)$ as chosen in the stage 1 of the reconstruction protocol. Next, we prove by induction that if the wire W evaluates to b then the players can compute $V(W, b)$. Clearly this is true of the input wires. If W is any other wire, it must be the output of an AND, OR, NOT, or $FANOUT$ gate G. In each of these cases, stage 2 of the reconstruction algorithm shows how the players can obtain the $V(W, b)$ from the relevant V-values of the input wires of G.

Security. Suppose $A \notin \mathcal{A}$. Then for every $P_i \notin A$, P has no way of knowing the 1-share of P_i's input wire. Therefore when P evaluates the circuit in stage 2, the input x_i must be 0. Hence from the definition of C there is no assignment $(y_1, y_2, \ldots y_n)$ which will make C evaluate to 1. Further, since C computes a monotone function, C will evaluate to zero even if some of the inputs x_i with $P_i \in A$ are set to zero.

It remains to prove the correctness of stage 2, i.e, that P cannot find the secret if C evaluates to 0. The proof of this is very similar to the proof of security of the reconstruction algorithm of the first protocol, and is hence omitted.

Security against the third party. In the sharing protocol, the T gets no information about the 1-shares of the input wires. Further, from the definition of OT, T gets no new information in stage 2 of the reconstruction protocol. Therefore, T can only evaluate the circuit with all inputs 0, which means that T cannot get the 1-share of the output wire.

Again, since T learns nothing at all during interaction with the virtual player, T cannot identify the access set A.

4.3 Efficiency

As with the previous protocol, this one is also efficient when the circuit is of polynomial size (i.e, $\mathcal{A} \in mNP$) since the total size of the shares is linear in n and the amount of published information is proportional to the number of $FANOUT$ gates. The question of computational complexity is somewhat tricky.

Strictly speaking, the protocol is computationally efficient since the sharing and reconstruction algorithms involve only a constant amount of computation for each gate of the circuit. However, the players need to determine if the set A is a qualified subset before they can start the reconstruction algorithm. This computation is, by definition, a general problem in mNP. The implications of this are discussed in the next section.

The *round complexity* of the interactive protocol between T and P is the same as the round complexity of the OT protocol used, since all the $m+n$ OTs can be invoked in parallel. If we use a scheme like the ones in [11], this complexity is 2. It might appear at first glance that if we use non-interactive OT schemes like [4], then the need for a third party would disappear. However, non-interactive OT schemes are not applicable in this context since the receiver needs to choose the index after the start of the protocol. Using non-interactive OT would require the access set of players to be known beforehand.

5 Discussion and Future Work

Theorem 3 (A simple upper bound). *Efficient CSS schemes cannot exist for an access structure $\mathcal{A}$ not in co-RP.*

Proof. To show this, we construct a deterministic algorithm to solve the problem "Does $A \in \mathcal{A}$" using the algorithms *share* and *reconstruct* as oracles. The algorithm chooses a random secret, shares it and uses *reconstruct* with the shares corresponding to A as inputs to see if it gets back the secret which it chose. If it is the same it decides that $A \in \mathcal{A}$. Else, it decides that $A \notin \mathcal{A}$. We note that if indeed $A \in \mathcal{A}$, it decides correctly with probability 1, while if $A \notin \mathcal{A}$, there is a small probability of error. Since we decide $L_{\mathcal{A}}$ with a deterministic algorithm, if *share* and *reconstruct* are poly-time then $\mathcal{A}$ must be in co-RP. □

Implications of Our Results

- It was not known whether it is possible to construct efficient secret sharing schemes for access structures outside (algebraic $NC^2\cap$ mono), though [2] provided evidence that it is possible. Our result shows that computational secret sharing is possible over the entire class mP which contains access structures not in algebraic NC^2.
- Combining our result with that of [1], it is possible to achieve the optimal information rate (for large secret length) for every access structure in mP.
- As we have remarked earlier, to carry out the reconstruction algorithm the players need to determine if they form an access set, and this computation could lie outside P in the third party case. This does not mean, however, that the third party result is purely of theoretical significance, for two reasons: when the players are probabilistic algorithms, the class of access structures that can be decided in poly-time is co-$RP\cap$ mono, as shown above, and this class is bigger than mP. Further, even for access structures admitting of deterministic poly-size circuits, it might be more efficient to use a randomized algorithm, in which case the protocol using nondeterministic Boolean circuits must be used.

Our result must be understood more as an existence result for efficient protocols, rather than as a method to construct such protocols. For instance, it is likely that using threshold gates as building blocks in addition to AND and OR would give more efficient protocols. One direction for future work in this area is to construct particularly efficient CSS schemes for interesting special cases of access structures in monotone P.

The most important open question is to determine the exact power of efficient CSS schemes; in particular, do there exist efficient CSS schemes for access structures outside monotone P.

Another direction for future work is to investigate models for CSS which do not have the co-RP upper bound. One way of doing this would be to allow the dealer to be computationally unbounded. This would have the effect of the language $L_{\mathcal{A}}$ being decidable in poly-time by a Turing machine with advice strings. However it is unreasonable to assume the dealer alone to be computationally unbounded, since secret sharing is usually a part of a larger protocol.

Another approach has been explored by Cachin [5], who uses a bulletin board on which an arbitrary amount of information may be published. Our construction using a semi-trusted third party also, as we have remarked, sidesteps the problem. We propose yet another approach. Since the trouble with access structures outside co-RP is the infeasibility of determining whether or not a given set is in the access structure, we provide the players with an oracle which can answer precisely that question: the players can make queries to the oracle with a subset of $\mathcal{P}$ as input, and the oracle decides whether it is a qualified subset or not. We note that this is a weaker assumption than that of [5]. It would be interesting to see if it is possible to construct efficient CSS schemes for the class mNP under this model.

References

1. P. Beguin and A. Cresti. General short computational secret sharing schemes. In *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *LNCS*, pp. 194–208, 1995.
2. A. Beimel and Y. Ishai. On the Power of Non-Linear Secret Sharing. newblock *In proceedings of 16th IEEE Structure in Complexity Theory* , 2001.
3. J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. *In proceedings of CRYPTO 88*, volume 403 of *LNCS*, pp. 27–36. Springer-Verlag, 1988.
4. M. Bellare and S. Micali. Non-interactive oblivous transfer and applications. *Advances in Cryptology - CRYPTO'89*, pp. 547–557, 1989.
5. Cachin. On-line secret sharing. In *IMA: IMA Conference on Cryptography and Coding, LNCS lately (earlier: Cryptography and Coding II, Clarendon Press, 1992)*, 1995.
6. M. K. Franklin and M. K. Reiter. Fair exchange with a semi-trusted third party. In *ACM Conference on Computer and Communications Security*, pp. 1–5, 1997.
7. Grigni and Sipser. Monotone Complexity. In *PATBOOL: Boolean Function Complexity*, London Mathematical Society Lecture Note Series 169, Cambridge University Press, 1992.
8. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proceedings of IEEE Globecom 87*, pp. 99–102. IEEE, 1987.
9. M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the 8th Annual IEEE Structure in Complexity Theory*, pp. 102–111, 1993.
10. H. Krawczyk. Secret sharing made short. In *CRYPTO'93*, LNCS, pp. 136–146, 1993.
11. M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *SODA 01*, pp. 448–457, 2001.
12. M. Rabin. How to exchange secrets by oblivious transfer. *Technical Report TR-81, Aiken Computation Laboratory, Harvard University*, 1981.
13. M. Rabin. Efficient dispersal of information for security, load-balancing and fault-tolerance. *JACM*, volume 36, pp. 335–348, 1989.
14. A. Razborov. A lower bound on the monotone network complexity of the logical permanent, (in russian). In *Mat. Zametki*, volume 37(6), pp. 887 – 900, 1985.
15. A. C. Yao. How to generate and exchange secrets. *In Proc. of STOC*, pp. 162–167, 1986.