A multilevel and low-locality BGW MPC protocol

Alexandra Berkoff[1], Benjamin Fuller[1,3], Sophia Yakoubov[2,4], and Arkady Yerukhimovich[1,5]

[1]affiliated with MIT Lincoln Laboratory at time of research
[2]MIT Lincoln Laboratory
[3]University of Connecticut
[4]Boston University
[5]George Washington University

Abstract

This work asks if the passive protocol of Ben-Or, Goldwasser, and Widgerson (STOC 1988) can be adapted for low communication locality. The core of the design is replacing the high degree initial secret sharing with a tree of constant degree sharings. This revised protocol can be adapted for correctness. The core question of the manuscript is whether the revised degree reduction protocol is secure for a constant fraction of malicious participants.

1 Low-locality through repeated secret sharing?

One of the emerging questions in MPC is the required communication graph between parties. The recent work of Boyle et al. [BCDH18] showed that in many natural circumstances the graph must be an expander. We work in the following model:

1. There is a single party P^* who seeks to have parties $P_1, ..., P_n$ perform a private computation. This party can send a single distribution message to all parties and receive a single reconstruction message from all parties.
2. We seek information theoretic security.
3. We assume a passive, static adversary that sees the view of at most a constant fraction of parties.
4. We assume that the communication graph between $P_1, ..., P_n$ can be reconfigured on the fly and that the complete graph is available for communication.

The core question of this paper is whether the traditional BGW [BOGW88] protocol can be adapted for low communication locality. We consider the passive protocol that uses Shamir secret sharing [Sha79] (secret sharing using fixed degree polynomials). We propose a modification to this protocol we call *levelled secret sharing* that we have not been able to show secure or insecure. The idea is as follows:

1. Rather than distributing a secret s among n parties by setting it as the zero of a degree $t = \Theta(n)$ polynomial, the initiating party P^* initiates a tree of sharings. Namely, P^* first sets s as the zero of a random *constant* degree polynomial, with a constant number of shares. For example, with constant degree 1 and 3 shares, P^* creates the shares $\langle s\rangle_1, \langle s\rangle_2, \langle s\rangle_3$. It then shares of each of *those* shares, so $\langle s\rangle_i$ would be the zero of a line with shares $\langle\langle s\rangle_i\rangle_1, \langle\langle s\rangle_i\rangle_2, \langle\langle s\rangle_i\rangle_3$. It repeats this process $\log(n)$ times until there are n shares at the lowest level of sharing. Each party receives exactly one of the lowest-level shares. An example with $n = 9$ parties and two levels is shown in Figure 1. Throughout this process the assignment of lowest-level shares to the n parties is entirely random but published to the parties following adversary choice of corruptions. (Alternatively, we can think of an adversary corrupting random parties.)
2. These n shares are distributed to the n computing parties.
3. Addition proceeds without communication, as in the original BGW protocol.
4. Multiplication uses the simplified degree reduction protocol for BGW designed by Gennaro, Rabin, and Rabin [GRR98], however, each level of polynomials is individually reduced. The key idea is that this should require each party to communicate with $3\log(n)$ parties (3 parties for each level of degree reduction).
5. At the end of the computation, the final shares are returned to P^*.

We can show this protocol is correct and that the initial sharing is secure. However, we have been unable to show that the modified degree reduction protocol preserves security. The majority of this manuscript is connecting this modified degree reduction protocol to a matrix problem that we have been unable to solve.

2 Review of the BGW Protocol

In the original protocol BGW protocol [BOGW88], there are n parties, $P_1, \ldots, P_n$, each of whom holds a secret value $s_1, \ldots, s_n$. Their goal is to run a protocol that jointly computes an arithmetic circuit f (over $\mathbb{Z}_q$) on their secret values while *still keeping their input values secret.*

Leveled Secret Sharing:

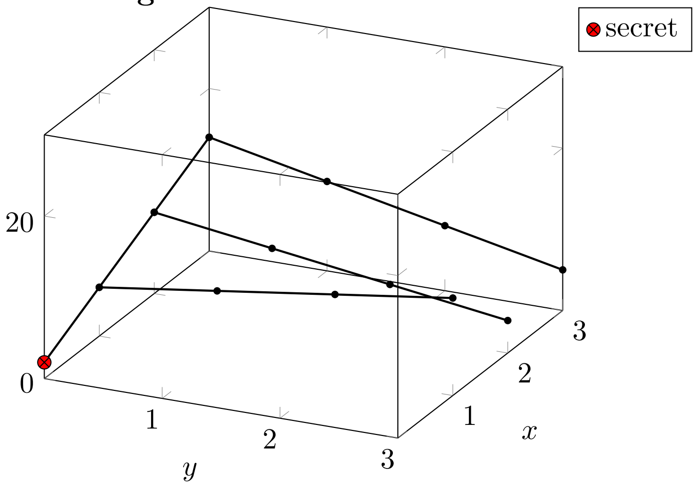


Figure 1: Two level sharing of initial secret.

To differentiate what can be learned from the *function* being computed versus what the *protocol* itself leaks, the security definition uses the real/ideal paradigm. Suppose we live in an ideal world, and there is a trusted party T to whom each P_i can send its s_i. T simply computes $f(s_1, \ldots, s_n)$ by itself and broadcasts the result to all parties. In the real world, of course, there is no trusted T. We consider a protocol secure if no party learns more in the real world than he would in the ideal world.

Adversary We assume that some subset of the parties are adversarial. For the purposes of this document, we are considering a fairly weak adversarial model, that perhaps we can strengthen in the future. Specifically, we are in the **threshold** model with a **static**, **semi-honest** adversary.

- **semi-honest**: The adversary does not maliciously change the messages sent or stop messages from being delivered. He simply observes the protocol and attempts to learn secrets he should not know. A decent analogy is that each party is a separate computer in a network, and the adversary has successfully gotten some spyware installed on a fraction of the machines. The spyware doesn't change how the machines behave, but it does log all of their behavior and report it all back to the adversary.
- **threshold**: We assume that there is a single adversary who controls up to a constant fraction of the parties. (Say, for concreteness, 1/6 of the parties.) In our spyware analogy, this means that even if the adversary can see the logs of 1/6 of all the machines in the network, he still can't learn anything about the honest parties'

secrets.

- **static**: The adversary cannot adaptively choose who to corrupt. Intuitively, an adaptive adversary could figure out who the "key players" in a protocol are by observing the protocol for a while, and then specifically decide to corrupt those people in order to do more damage. In our model, the adversary has to decide who to corrupt *before* the protocol begins. In fact, in our protocol, no party gets its role assigned until after corruptions have taken place, so an adversary's best strategy is to randomly corrupt 1/6 of the parties.

2.1 How the protocol works

In the BGW protocol there are three major steps:

1. First, each party P_i "shares" its value s_i in a secure way so that each other party P_j has a share of P_i's secret. We denote this $\langle s_i \rangle_j$.
2. Each party locally computes the circuit (arithmetic addition and multiplication gates) on the shares of the secrets, communicating with others only when something cannot be done locally.
3. Finally, the parties broadcast their share of the final answer to everyone else, and each party locally reconstructs the answer from the shares he has seen.

Secret Sharing The BGW protocol uses Shamir secret sharing [Sha81], which works as follows: Party P_i shares a secret $s_i \in \mathbb{Z}_q$ by choosing $a_{i1}, a_{i2}, \ldots, a_{it}$ uniformly at random in $\mathbb{Z}_q$, and setting $f_i(x) = s_i + a_{i1}x + a_{i2}x^2 + \ldots + a_{it}x^t$. For each $j \neq i$, he sends $f_i(j)$ (which we denote $\langle s_i \rangle_j$) to party P_j. This sharing method relies heavily on the fact that any $t + 1$ points uniquely define a degree t polynomial.

Lemma 1 (Lagrange Interpolation)**.** *Given a polynomial $f(x)$ with degree at most t, and $t + 1$ points on that polynomial $(x_1, f(x_1)), (x_2, f(x_2)), \ldots (x_{t+1}, f(x_{t+1}))$ the coefficients for that polynomial can be derived via the following expression:*

$$f(x) = \sum_{j=1}^{t+1} f(x_j)\ell_j(x)$$

where

$$\ell_j = \frac{\prod_{i\neq j}(x - x_i)}{\prod_{i\neq j}(x_j - x_i)}$$

The polynomials ℓ_j are called the fundamental Lagrange polynomials.

So, for any set of parties $\mathcal{Q} \subseteq \{P_1, \ldots, P_n\}$ such that $|\mathcal{Q}| \geq t+1$, to learn a secret $s_i = f_i(0)$, the parties in $\mathcal{Q}$ simply send their shares to each other and compute

$$\sum_{P_j \in \mathcal{Q}} \langle s_i \rangle_j \ell_j(0)$$

If, on the other hand, the adversary $\mathcal{A}$ controls some set of parties $\mathcal{Q}_\mathcal{A}$ such that $|\mathcal{Q}_\mathcal{A}| \leq t$, even when $\mathcal{A}$ knows $f_i(j)$ for each $P_j \in \mathcal{Q}_\mathcal{A}$, there are many possible different polynomials $f'(x)$ where $f'(j) = f_i(j)$ for $p_j \in \mathcal{Q}_\mathcal{A}$. Call this set of polynomials $\mathcal{F}$. Because the coefficients of f_i were chosen uniformly at random, we can show that the value $f'(0)$ for randomly chosen $f' \in \mathcal{F}$ is uniformly distributed over $\mathbb{Z}_q$, so the secret s_i is totally hidden.

Addition Addition is really simple. For party P_j to get a share of $s_1 + s_2$, he simply locally computes $\langle s_1 \rangle_j + \langle s_2 \rangle_j$. This gives him the value $f_1(j) + f_2(j)$. If we define $f_{sum}(x) = f_1(x) + f_2(x)$, then P_j now has a share on $f_{sum}(x)$, a degree t polynomial with uniformly distributed coefficients whose zero is $s_1 + s_2$.

Multiplication Multiplication cannot be done locally in the same way. If P_j wanted to obtain a share of $s_1 \times s_2$, he could locally compute $\langle s_1 \rangle_j \times \langle s_2 \rangle_j$, which would give him a share on the polynomial $f_{prod}(x) = f_1(x) \times f_2(x)$, and in fact $f_{prod}(0) = s_1 \times s_2$. However, f_{prod} will be of degree $2t$ instead of t, meaning it will take twice as many parties to recover the secret. Note that each multiplication would further grow the degree of the underlying polynomial. The degree could quickly grow larger than the total number of parties, making the secret unrecoverable. Furthermore, the coefficients of f_{prod} are not uniformly distributed, which is important for proving that from the point of view of the adversary, every secret value is equally likely.

The solution is to run a protocol that simultaneously re-randomizes coefficients and reduces degree[1]. This protocol will, for all j, change $\langle s_1 \times s_2 \rangle_j$ from $f_{prod}(j)$, a point on a degree $2t$ polynomial, to $g(j)$, a point on a new, random, degree t polynomial such that $f_{prod}(0) = g(0)$. In the following, $\ell_j(0)$ refers to the fundamental Lagrange polynomial described in Lemma 1, evaluated at 0. The protocol proceeds as follows:

Degree Reduction[GRR98] For $i = 1, \ldots n$

[1] In the original BGW paper, the parties first run a re-randomization protocol and then perform degree reduction. We present the modified protocol described by Gennaro, Rabin, and Rabin that combines re-randomization and degree reduction into a single step.[GRR98]

- P_i generates a new random, degree t polynomial $g_i(x)$ such that $g_i(0) = f_{prod}(i)$, and all other coefficients are chosen uniformly at random from $\mathbb{Z}_q$.
- For $j = 1, \ldots, n$, P_i sends $g_i(j)$ to P_j, and receives $g_j(i)$ from P_j.
- P_i computes $\sum_{j=1}^{n} \ell_j(0) g_j(i)$.

If we define $g(x) = \sum_{j=1}^{n} \ell_j(0) g_j(x)$, then clearly for all i, P_i has the share $g(i)$. Furthermore, since $g(x)$ is a linear combination of random degree t polynomials, it is itself a random degree t polynomial. Finally,

$$g(0) = \sum_{j=1}^{n} \ell_j(0) g_j(0) = \sum_{j=1}^{n} \ell_j(0) f_{prod}(j) = f_{prod}(0)$$

3 Our Protocol - Two Levels

You can think of our protocol as basically a "leveled" extension of the BGW protocol. As a toy example, suppose we have 9 parties, and instead of labeling them $P_1, \ldots, P_9$, we label them $P_{i,j}$ for $i, j \in \{1, 2, 3\}$.

To share a secret s, first choose a linear $f(x) = ax + s$, and then for $i = 1, 2, 3$, choose $f_i(y) = a_i y + f(i)$, where all the a coefficients are chosen at random. Each party $P_{i,j}$ gets the share $\langle s \rangle_{i,j} = f_i(j)$.

3.1 Addition and Multiplication

Figure 2 shows example sharing structures of two separate secrets s_1 and s_2, where we describe the sharing structure of s_1 using $f(x)$ s.t. $f(0) = s_1$, and $f_i(y)$ s.t. $f_i(0) = f(i)$, for $i = 1, 2, 3$, and similarly describe the sharing structure of s_2 using $g(x)$ s.t. $g(0) = s_2$ and three $g_i(y)$ s.t. $g_i(0) = g(i)$, for $i = 1, 2, 3$. Each party $P_{i,j}$ has a share $\langle s_1 \rangle_{i,j} = f_i(j)$ and $\langle s_2 \rangle_{i,j} = g_i(j)$. To add, just as in the original BGW protocol, each party $P_{i,j}$ simply computes

$$\langle s_1 + s_2 \rangle_{i,j} = \langle s_1 \rangle_{i,j} + \langle s_2 \rangle_{i,j}$$

To multiply, each party $P_{i,j}$ computes

$$\langle s_1 \cdot s_2 \rangle_{i,j} = \langle s_1 \rangle_{i,j} \cdot \langle s_2 \rangle_{i,j}$$

and then participates in a degree reduction protocol. It is this degree reduction protocol that gives us lower communication locality.

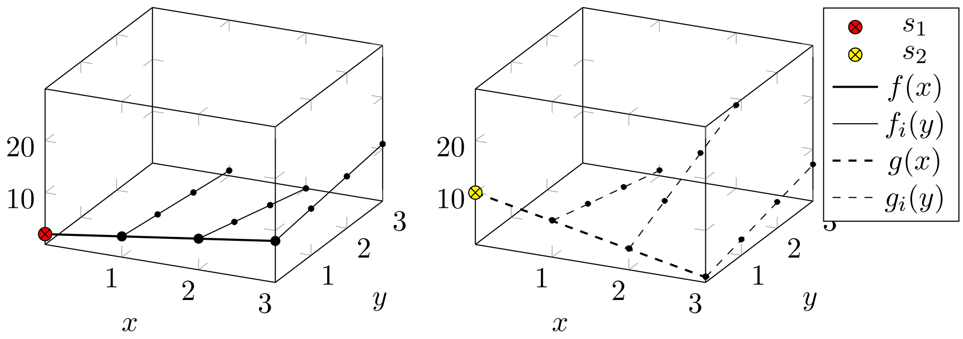


Figure 2: Leveled secret sharings of the secrets s_1 and s_2

3.2 Degree Reduction

After each party has multiplied its two shares together, the sharing structure consists of a quadratic $f \cdot g(x) = f(x) \cdot g(x)$ such that $f \cdot g(0) = s_1 \cdot s_2$, and three quadratics $f_i \cdot g_i(y) = f_i(y) \cdot g_i(y)$ such that $f_i \cdot g_i(0) = f \cdot g(i)$.

Note, however, that for a fixed j, the points $f_1 \cdot g_1(j)$, $f_2 \cdot g_2(j)$ and $f_3 \cdot g_3(j)$ also implicitly define a quadratic equation. Call this equation $h'_j(x)$. Furthermore, denoting $h'_j(0) = z_j$ for each j, the three points z_1, z_2, and z_3 implicitly define a quadratic we'll call $h'(y)$, and it can be shown that $h'(0) = s_1 \cdot s_2$. This relationship between the implicit and explicit polynomials is illustrated in Figure 3. Now, rather than communicating with all of the parties,

1. First, the parties perform BGW degree reduction (described in section 2.1) with the parties they share a j coordinate with. More specifically, $P_{i,j}$ runs BGW degree reduction with the parties $P_{i',j}$ for $i' = 1, 2, 3$, replacing the implicit quadratic $h'_j(x)$ with a linear $h_j(x)$, such that $h_j(0) = h'_j(0)$. For $j = 1, 2, 3$, letting $\ell_j(0)$ denote the Lagrange coefficient from Lemma 1, we define the linear equation

$$h(x) = \sum_{j=1}^{3} \ell_j(0) h_j(x)$$

   This is illustrated in Figure 4.

2. Next, noting that for $i = 1, 2, 3$, the values $h_1(i)$, $h_2(i)$ and $h_3(i)$ implicitly form the quadratics $h_i(y)$, we run BGW degree reduction on those quadratics. More specifically, $P_{i,j}$ runs the BGW protocol with $P_{i,j'}$ for $j' = 1, 2, 3$. This gives her a share $h_i(j)$ on a *new* random linear polynomial $h_i(y)$. Furthermore, it is straightforward to show that $h_i(0) = h(0)$ as defined above. As illustrated in Figure 5, this completes the degree reduction process.

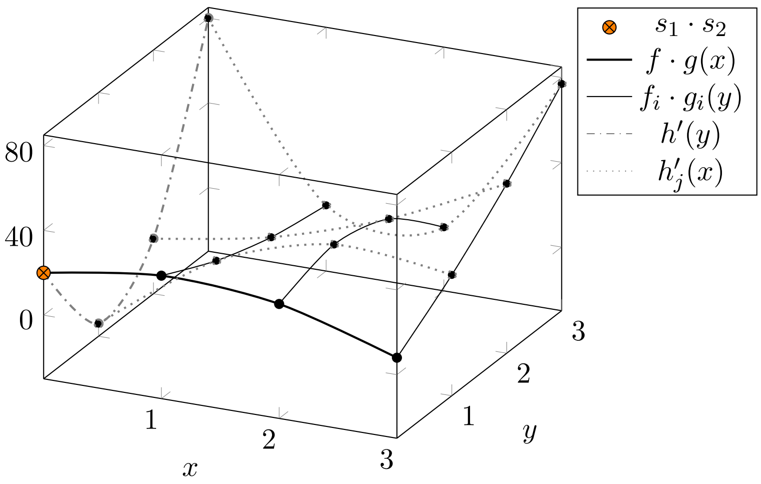


Figure 3: The multiplied secret sharing of $s_1 \times s_2$

We stress that degree reduction is done backwards, with the top sharing being degree reduced first followed by the second sharing.

4 Many Levels

In our protocol, we extend the logic of the two-level example as described below.

Notation Let $n = 3^d$ be the number of parties. Each party is labeled by a vector $\mathbf{i} \in \{1,2,3\}^d$. Let $\mathbf{i}[j,k]$ denote the j^{th} through k^{th} elements of $\mathbf{i}$. If $j > k$, then let $\mathbf{i}[j,k]$ denote the empty vector.

A secret is initially shared on a tree of polynomials illustrated below, where the invariant is that $f^{(0)}_{\mathbf{i}[1,\ell]}(0) = f^{(0)}_{\mathbf{i}[1,\ell-1]}(\ell)$, and that party $\mathbf{i}$ holds the value $f^{(0)}_{\mathbf{i}[1,d-1]}(\mathbf{i}[d])$.

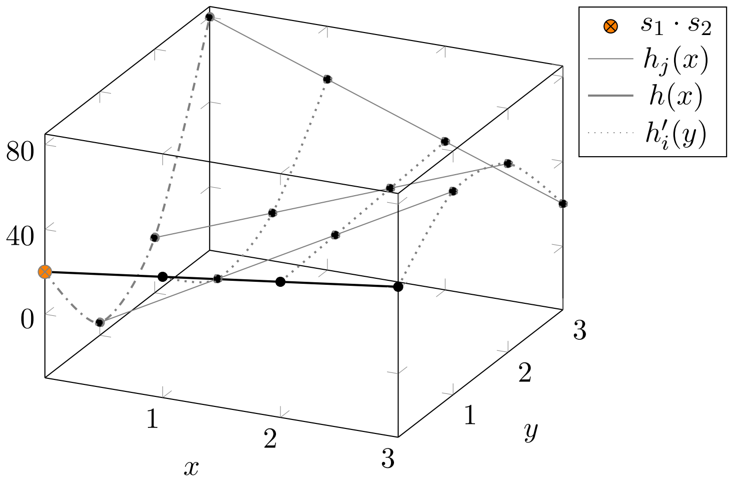


Figure 4: Sharing structure after the first step of degree reduction.

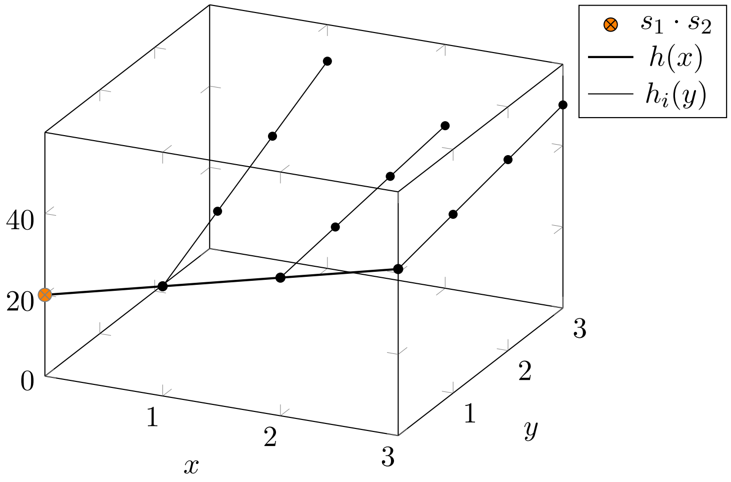
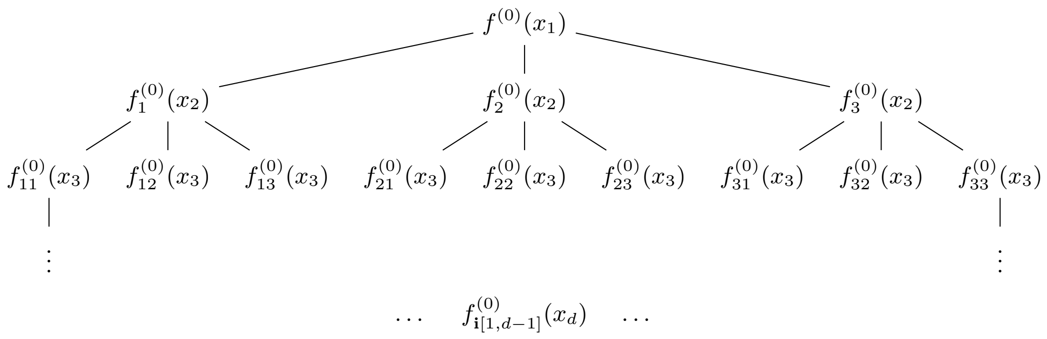


Figure 5: Sharing structure after the second step of degree reduction.



The Protocol We focus our discussion on the degree reduction protocol, sharing, reconstruction, and addition naturally extend from the two level case. The degree reduction protocol proceeds in d rounds. We denote values specific to round r with the superscript (r). So, for example, in round r, party $\mathbf{i}$ is a member of quorum $\mathcal{Q}^{(r)}(\mathbf{i}) = \{\text{parties } \hat{\mathbf{i}} \text{ s.t. } \hat{\mathbf{i}} \setminus \hat{\mathbf{i}}[r] = \mathbf{i} \setminus \mathbf{i}[r]\}$. Denote the secret that party $\mathbf{i}$ holds in round 0 (before the start of the protocol) as: $s_{\mathbf{i}}^{(0)}$.

For rounds $r = 1, \dots, d$:

1. Each party $\mathbf{i}$ chooses a new degree 1 polynomial $f^{(r)}_{\mathbf{i}[r,d],\mathbf{i}[1,r-1]}$, such that

$$f^{(r)}_{\mathbf{i}[r,d],\mathbf{i}[1,r-1]}(0) = s_{\mathbf{i}}^{(r-1)} \tag{1}$$

2. Next, for each party $\hat{\mathbf{i}}$ in $\mathcal{Q}^{(r)}(\mathbf{i})$, Party $\mathbf{i}$ evaluates $f^{(r)}_{\mathbf{i}[r,d],\mathbf{i}[1,r-1]}(\hat{\mathbf{i}}[r])$ and sends the value to $\hat{\mathbf{i}}$.

3. Now, party $\mathbf{i}$ has the values $f^{(r)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(\mathbf{i}[r])$, for $j \in \{\hat{\mathbf{i}}[r] \text{ s.t. } \hat{\mathbf{i}} \in \mathcal{Q}^{(r)}(\mathbf{i})\}$.
   He computes:

$$f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(\mathbf{i}[r]) = \sum_{j=1}^{3} \ell_j(0) f^{(r)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(\mathbf{i}[r]) \tag{2}$$

4. Finally, party $\mathbf{i}$ sets $s_{\mathbf{i}}^{(r)} = f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(\mathbf{i}[r])$.

At the end of the degree reduction protocol, the final share of party $\mathbf{i}$ is $s_{\mathbf{i}}^{(d)}$.

4.1 The Polynomials

Each polynomial $f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(x_r)$ is freshly generated in step r of the degree reduction protocol, but its zero is on a polynomial that is the linear combination of three polynomials from round $r-1$. with new, random coefficients, but notice that if we define:

$$f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-2]}(x_{r-1}) = \sum_{j=1}^{3} \ell_j(0) f^{(r-1)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-2]}(x_{r-1})$$

we can show that

$$f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(0) = f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-2]}(\mathbf{i}[r-1])$$

The algebra is:

$$\begin{aligned}
f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(0) &= \sum_{k=1}^{3} \ell_k(0) f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(k) \\
&= \sum_{k=1}^{3} \ell_k(0) \sum_{j=1}^{3} \ell_j(0) f^{(r)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(k) \text{ as defined in Eq (2)} \\
&= \sum_{j=1}^{3} \ell_j(0) \sum_{k=1}^{3} \ell_k(0) f^{(r)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(k) \\
&= \sum_{j=1}^{3} \ell_j(0) f^{(r)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(0) \text{ by the definition of interpolation} \\
&= \sum_{j=1}^{3} \ell_j(0) f^{(r-1)}_{j,\mathbf{i}[r+1,d],\mathbf{i}[1,r-2]}(\mathbf{i}[r-1]) \text{ as defined in Eq (1)} \\
&= f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-2]}(\mathbf{i}[r-1])
\end{aligned}$$

In general, this relationship between rounds (level ℓ polynomials in round $r-1$ combine to make level $\ell-1$ polynomials in round r) holds throughout the degree reduction protocol. The sharing structure for each round is described below.

Round 0	Round 1	Round 2	$\dots$	Round r	$\dots$	Round $d-1$	Round d
$f^{(0)}_{\mathbf{i}[1,0]}(x_1)$	$f^{(1)}_{\mathbf{i}[2,1]}(x_2)$	$f^{(2)}_{\mathbf{i}[3,2]}(x_3)$	$\dots$	$f^{(r)}_{\mathbf{i}[r+1,r]}(x_{r+1})$	$\dots$	$f^{(d-1)}_{\mathbf{i}[d,d-1]}(x_d)$	$f^{(d)}_{\mathbf{i}[1,0]}(x_1)$
$f^{(0)}_{\mathbf{i}[1,1]}(x_2)$	$f^{(1)}_{\mathbf{i}[2,2]}(x_3)$					$f^{(d-1)}_{\mathbf{i}[d,d]}(x_1)$	$f^{(d)}_{\mathbf{i}[1,1]}(x_2)$
$f^{(0)}_{\mathbf{i}[1,2]}(x_3)$				$\vdots$		$f^{(d-1)}_{\mathbf{i}[d],\mathbf{i}[1]}(x_2)$	
				$f^{(r)}_{\mathbf{i}[r+1,d-1]}(x_d)$			
			$\iddots$	$f^{(r)}_{\mathbf{i}[r+1,d]}(x_1)$	$\iddots$		
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
				$f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,\ell]}(x_{\ell+1})$			
		$f^{(2)}_{\mathbf{i}[3,d-1]}(x_d)$					$f^{(d)}_{\mathbf{i}[1,d-3]}(x_{d-2})$
	$f^{(1)}_{\mathbf{i}[2,d-1]}(x_d)$	$f^{(2)}_{\mathbf{i}[3,d]}(x_1)$		$\vdots$		$f^{(d-1)}_{\mathbf{i}[d],\mathbf{i}[1,d-3]}(x_{d-2})$	$f^{(d)}_{\mathbf{i}[1,d-2]}(x_{d-1})$
$f^{(0)}_{\mathbf{i}[1,d-1]}(x_d)$	$f^{(1)}_{\mathbf{i}[2,d]}(x_1)$	$f^{(2)}_{\mathbf{i}[3,d],\mathbf{i}[1,1]}(x_2)$	$\dots$	$f^{(r)}_{\mathbf{i}[r+1,d],\mathbf{i}[1,r-1]}(x_r)$	$\dots$	$f^{(d-1)}_{\mathbf{i}[d],\mathbf{i}[1,d-2]}(x_{d-1})$	$f^{(d)}_{\mathbf{i}[1,d-1]}(x_d)$

5 When does $\mathcal{A}$ win?

The question we are trying to answer is: If an adversary $\mathcal{A}$ chooses a random subset of the parties to corrupt (such that this set of parties is a constant fraction of n) what is the probability that he recovers the secret?

Note that if it we were just talking about leveled secret sharing, and not about our complicated degree reduction protocol, this would be straightforward to analyze.

Analyzing the initial sharing Suppose all $\mathcal{A}$ has seen is a fresh leveled sharing of one secret. Instead of having $\mathcal{A}$ corrupt exactly $\frac{1}{6}$ of the players, it suffices to analyze the game where $\mathcal{A}$ corrupts each party independently with probability $\frac{1}{3}$. Then, with overwhelming probability, by the Chernoff bound, he corrupts at *least* $\frac{1}{6}$ of the players, meaning his probability of winning in this game is *better* than his probability in winning in the game where he corrupts exactly $\frac{1}{6}$ of the players. Thus, if we can show he still has an exponentially small probability of winning in this game, we are done.

In this game, $\mathcal{A}$ recovers $f_{\mathbf{i}[1,d-1]}(0)$ if he knows at least two of $f_{\mathbf{i}[1,d-1]}(1)$ $f_{\mathbf{i}[1,d-1]}(2)$ and $f_{\mathbf{i}[1,d-1]}(3)$. In general, he recovers $f_{\mathbf{i}[1,\ell]}(0)$ if he has recovered at least two of $f_{\mathbf{i}[1,\ell]}(1)$, $f_{\mathbf{i}[1,\ell]}(2)$, and $f_{\mathbf{i}[1,\ell]}(3)$. Recovering the secret just means recovering $f(0)$.

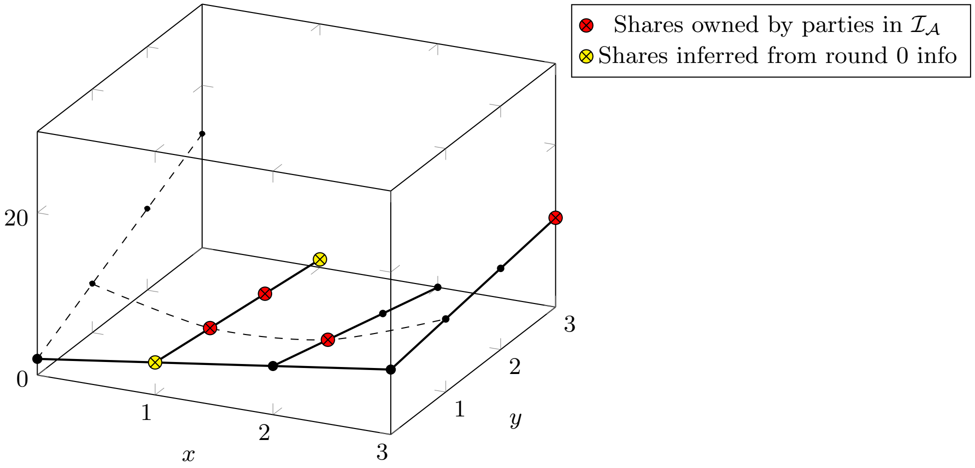
In this scenario, $\mathcal{A}$'s probability of recovering a $f_{\mathbf{i}[1,d-1]}(0)$ is $\leq \frac{1}{3}^2$. In general, letting p_ℓ denote his probability of recovering a level ℓ zero, we have that $p_\ell \leq p_{\ell+1}^2$. So, $p_0 \leq \frac{1}{3}^{2^d} = \theta(\frac{1}{2^n})$.

5.1 Learning from Degree Reduction

The problem is that $\mathcal{A}$ may use information from multiple rounds of the degree reduction protocol to learn more than they could know in any one round. For simplicity of analysis, suppose that we had parties engaging in a degree reduction protocol on a fresh leveled sharing (degree 1 in each dimension). Clearly the adversary learns at least as much in this setting as he would in the setting where the degree reduction protocol was run on a degree-2 leveled sharing. Below, we illustrate the first step of the degree reduction protocol and demonstrate why even though $\mathcal{A}$ cannot recover the secret on a fresh sharing, he can after the degree reduction protocol.

Let's say we have 9 parties. There is a polynomial $f^{(0)}(x)$ such that $f^{(0)}(0) =$ secret, and three polynomials $f_i^{(0)}(y)$ such that $f_i^{(0)}(0) = f^{(0)}(i)$. Party (i,j) holds the share $f_i^{(0)}(j)$. Suppose the adversary, $\mathcal{A}$ controls the parties $\mathcal{I}_\mathcal{A} = \{(1,1),(2,1),(1,2),(3,3)\}$.

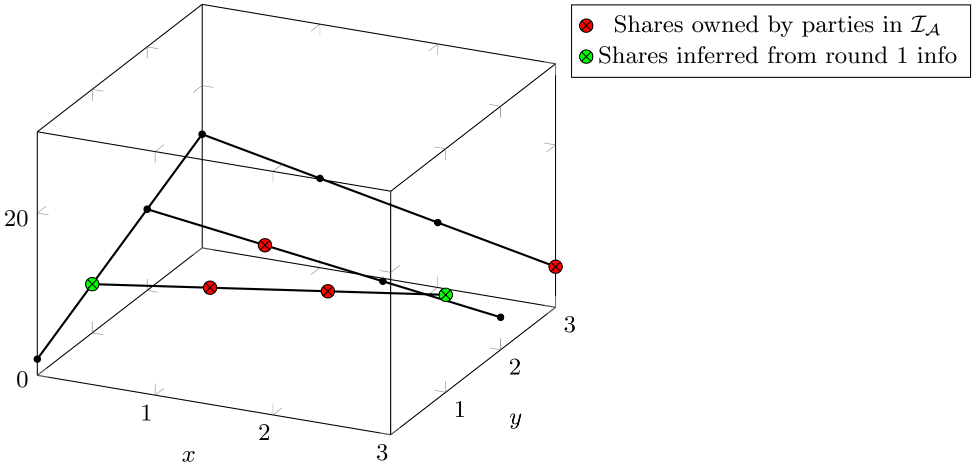
What $\mathcal{A}$ learns in Round 0:



⊗ Shares owned by parties in $\mathcal{I}_\mathcal{A}$
⊗ Shares inferred from round 0 info
20
0
1
2
3
x
1
2
3
y

Before degree reduction (round 0) because $\boxed{(1,1)}$ and $\boxed{(2,1)}$ both have shares on $f_1^{(0)}(y)$, $\mathcal{A}$ can learn $\boxed{f_1^{(0)}(0) = f^{(0)}(1)}$ (and also the share held by $\boxed{(1,3)}$).

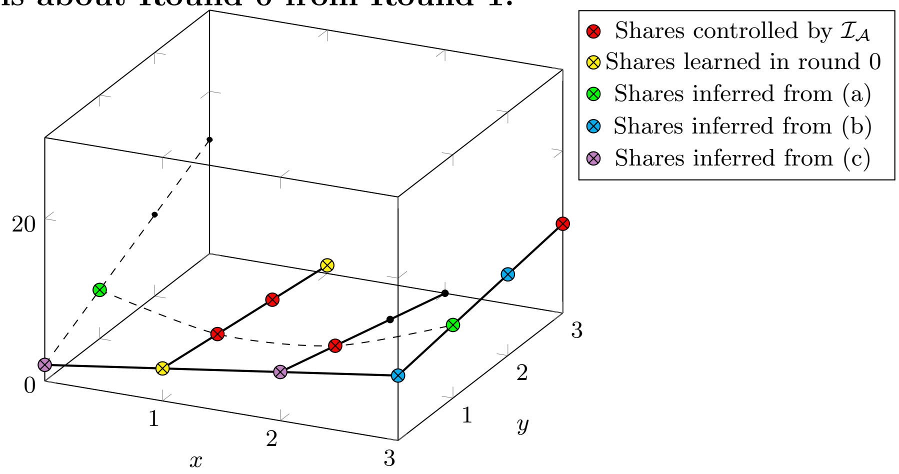
What $\mathcal{A}$ Learns about Round 1 in Round 1:



⊗ Shares owned by parties in $\mathcal{I}_\mathcal{A}$
⊗ Shares inferred from round 1 info
20
0
1
2
3
x
1
2
3
y

In step 1 of degree reduction because $\boxed{(1,1)}$ and $\boxed{(2,1)}$ lie on a line (call it $f_1^{(1)}(x)$), $\mathcal{A}$ can recover $\boxed{f_1^{(1)}(0)}$. Because $(3,1)$ shares its value from round 0 with a majority-bad quorum, $\mathcal{A}$ also learns $\boxed{f_1^{(1)}(3)}$ and $\boxed{f_3^{(0)}(1)}$. As we shall see, however, just knowing $f_1^{(1)}(0)$ is enough for $\mathcal{A}$ to calculate $f_3^{(0)}(1)$ directly.

What $\mathcal{A}$ learns about Round 0 from Round 1:



(a) Back in round 0, there was an implicit degree-2 polynomial, call it $\boxed{g_1^{(0)}(x)}$, defined by the points $f_1^{(0)}(1)$, $f_2^{(0)}(1)$, $f_3^{(0)}(1)$, and its zero is *the same as* $\boxed{f_1^{(1)}(0)}$. So, in fact, $\mathcal{A}$ knows $\boxed{g_1^{(0)}(0)}$. Since it also knows $\boxed{f_1^{(0)}(1) = g_1^{(0)}(2)}$ and $\boxed{f_2^{(0)}(1) = g_1^{(0)}(2)}$, it now has three points on $g_1^{(0)}(x)$, and can interpolate to learn $\boxed{g_1^{(0)}(3) = f_3^{(0)}(1)}$.

(b) Once $\mathcal{A}$ has recovered $\boxed{f_3^{(0)}(1)}$, since it already knows $\boxed{f_3^{(0)}(3)}$, it can recover $\boxed{f_3^{(0)}(2)}$ and $\boxed{f_3^{(0)}(0) = f^{(0)}(3)}$.

(c) And finally, now that $\mathcal{A}$ has $\boxed{f^{(0)}(3)}$ and $\boxed{f^{(0)}(1)}$, it can recover $\boxed{f^{(0)}(2)}$ and $\boxed{f^{(0)}(0) = \text{the secret.}}$

To summarize the issue, as we degree reduce this creates more "directions" that have low degree polynomials and allow the adversary to learn more shares than allowed by the initial sharing. The question we have is if for large enough n and d, the adversary will still be able to recover secrets with any noticeable probability.

6 Viewing this as a matrix problem.

It seems the cleanest way to understand this is as a matrix problem. Given $n = 3^d$ parties, each labeled by a vector $\mathbf{i} \in \{1, 2, 3\}^d$, a d-leveled sharing of a secret s among 3^d parties consists of $\sum_{j=0}^{d-1} 3^j$ polynomials over $\mathbb{Z}_q$:

- $f(x_1) = ax_1 + s$
- $f_{\mathbf{i}[1]}(x_2) = a_{\mathbf{i}[1]}x_2 + f(\mathbf{i}[1])$
- $f_{\mathbf{i}[1,2]}(x_3) = a_{\mathbf{i}[1,2]}x_3 + f_{\mathbf{i}[1]}(\mathbf{i}[2])$
- $\vdots$
- $f_{\mathbf{i}[1,d-1]}(x_d) = a_{\mathbf{i}[1,d-1]}x_d + f_{\mathbf{i}[1,d-2]}(\mathbf{i}[d-1])$

We denote the share party $\mathbf{i}$ holds as $\langle s\rangle_{\mathbf{i}}$. Its value is $f_{\mathbf{i}[1,d-1]}(\mathbf{i}[d])$.

In the above, each a coefficient is chosen uniformly at random from $\mathbb{Z}_q$. In some sense, the randomness from these a coefficients "hides" the value of s, our secret. Indeed, the share held by a party $\mathbf{i}$ can be written as a linear combination of the random a coefficients and the secret:

$$\langle s\rangle_{\mathbf{i}} = a \cdot \mathbf{i}[1] + a_{\mathbf{i}[1]} \cdot \mathbf{i}[2] + a_{\mathbf{i}[1,2]} \cdot \mathbf{i}[3] + \ldots + a_{\mathbf{i}[1,d-1]} \cdot \mathbf{i}[d]$$

So, for example, setting $d = 3$, we have a system of linear equations, as illustrated in Figure 6. We've highlighted a random $\frac{1}{3}$ of the rows, representing the information an adversary would know if he controlled the parties corresponding to those rows.

(In this example: $\mathcal{Q}_{\mathcal{A}} = \{P_{111}, P_{112}, P_{131}, P_{132}, P_{212}, P_{223}, P_{231}, P_{312}, P_{332}\}$)

In Figure 7, we consider just the matrix of rows the adversary controls. We've highlighted in blue the variables that are actually constrained by the information the adversary has, together with the corresponding columns.

Now, recovering s reduces to simply solving for s in the above system of linear equations. Given only the information corresponding to the blue submatrix, an adversary $\mathcal{A}$ can recover s if and only if it is of full rank. It is easy to see that the above submatrix has 12 columns, but only 9 rows, so it cannot possibly be of rank 12. The issue is, the adversary gets additional information after degree reduction.

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\
1 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
1 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3
\end{pmatrix}
\begin{pmatrix}
s \\ a \\ a_1 \\ a_2 \\ a_3 \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \\ a_{33}
\end{pmatrix}
=
\begin{pmatrix}
\langle s \rangle_{111} \\ \langle s \rangle_{112} \\ \langle s \rangle_{113} \\ \langle s \rangle_{121} \\ \langle s \rangle_{122} \\ \langle s \rangle_{123} \\ \langle s \rangle_{131} \\ \langle s \rangle_{132} \\ \langle s \rangle_{133} \\ \langle s \rangle_{211} \\ \langle s \rangle_{212} \\ \langle s \rangle_{213} \\ \langle s \rangle_{221} \\ \langle s \rangle_{222} \\ \langle s \rangle_{223} \\ \langle s \rangle_{231} \\ \langle s \rangle_{232} \\ \langle s \rangle_{233} \\ \langle s \rangle_{311} \\ \langle s \rangle_{312} \\ \langle s \rangle_{313} \\ \langle s \rangle_{321} \\ \langle s \rangle_{322} \\ \langle s \rangle_{323} \\ \langle s \rangle_{331} \\ \langle s \rangle_{332} \\ \langle s \rangle_{333}
\end{pmatrix}$$

Figure 6: System of Linear Equations in the Original Sharing

$$
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 3 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
1 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
s \\ a \\ a_1 \\ a_2 \\ a_3 \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \\ a_{33}
\end{pmatrix}
=
\begin{pmatrix}
\langle s \rangle_{111} \\ \langle s \rangle_{112} \\ \langle s \rangle_{131} \\ \langle s \rangle_{132} \\ \langle s \rangle_{212} \\ \langle s \rangle_{223} \\ \langle s \rangle_{231} \\ \langle s \rangle_{312} \\ \langle s \rangle_{332}
\end{pmatrix}
$$

Figure 7: What the Adversary Knows in Round 0

6.1 Multiplication/Degree Reduction

Recall that after the first round of degree reduction, the parties hold shares on polynomials:

- $f^{(1)}(x_2) = a^{(1)}x_2 + s$
- $f^{(1)}_{\mathbf{i}[2,2]}(x_3) = a^{(1)}_{\mathbf{i}[2,2]}x_2 + f(\mathbf{i}[1])$
- $\vdots$
- $f^{(1)}_{\mathbf{i}[2,d-1]}(x_d) = a^{(1)}_{\mathbf{i}[2,d-1]}x_d + f_{\mathbf{i}[2,d-2]}(\mathbf{i}[d-1])$
- $f^{(1)}_{\mathbf{i}[2,d]}(x_1) = a^{(1)}_{\mathbf{i}[2,d]}x_1 + f_{\mathbf{i}[2,d-1]}(\mathbf{i}[d])$

The coefficients $a^{(1)}_{\mathbf{i}[2,d]}$ are chosen uniformly at random, but the "higher level" coefficients are linear combinations of those from the previous round. For example, the coefficients $a^{(1)}_{\mathbf{i}[2,d-1]} = \sum_{j=1}^{3} \ell_j(0)a_{j,\mathbf{i}[2,d-1]}$. So, in the 3 level case, when going from "round 0" (before degree reduction has started) to round 1 (after the 1st step of degree reduction), the relationships between variables can be described by a series of 4 "linker rows" shown in Figure 8.

Also, notice that in round 1, party $\mathbf{i}$ now holds the share $f^{(1)}_{\mathbf{i}[2,d]}(\mathbf{i}[i])$ So in our running example, as illustrated in Figure 9, the shares $\mathcal{A}$ knows are: $\{\langle s\rangle^{(1)}_{111}, \langle s\rangle^{(1)}_{121}, \langle s\rangle^{(1)}_{311}, \langle s\rangle^{(1)}_{321}, \langle s\rangle^{(1)}_{122}, \langle s\rangle^{(1)}_{232}, \langle s\rangle^{(1)}_{312}, \langle s\rangle^{(1)}_{123}, \langle s\rangle^{(1)}_{323}\}$.

In essence, the adversary gets to see a different blue submatrix each round, and furthermore, has access to "linker rows" (information relating one round of degree reduction to the next). If the adversary can find *any* full rank submatrix among all of this information, he recovers s.

Let R_r be the matrix corresponding to the shares that $\mathcal{A}$ learns during round r of degree reduction. Note that if the adversary corrupts $\frac{1}{3}$ of the parties, then each R_r has exactly $\frac{1}{3}n = 3^{d-1}$ rows. Also, not counting the variable s, which appears in R_r for all r, there are $\sum_{i=1}^{d-1} 3^i$ new variables per round, so each R_r has $\sum_{i=1}^{d-1} 3^i$ columns, of which some number $z \geq 0$ are all-zeroes. Let R_{all} be the matrix consisting of every R_r for $r = 0, \ldots, d$.

Let $L_{r,r+1}$ be the matrix corresponding to the linker rows between rounds r and $r+1$. Note that there are $\sum_{i=0}^{d-2} 3^i$ linker rows between every two rounds. Similarly, let L_{all} be the matrix consisting of all linker rows. Figure 11 is a visualization of R_{all} and Figure 12 is a visualization of L_{all}.

Now, the probability that $\mathcal{A}$ recovers the secret is simply the probability that there exists R', a subset of the rows of R_{all} and L', a subset of the rows of L_{all} such that, letting $A = R' \cup L'$, the rank of A is equal to the number of non-zero columns of A.

How do we calculate this probability? It is unclear to us, as d grows, whether it increases or decreases. How do we even find R'? It seems that especially for large d, R' may include strict subsets of the rows of R_r for each r. For example, in our 3-level example from before, if we removed the 5th row of R_1, we would decrease the rank by 1, but we would decrease the number of non-zero columns by 2. So that's where we're stuck. We'd love help and ideas from the community!

References

[BCDH18] Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. Must the communication graph of mpc protocols be an expander? In *Annual International Cryptology Conference*, pages 243–272. Springer, 2018.

$$\begin{pmatrix} 0 & 0 & 3 & -3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} s \\ a^{(0)} \\ a_1^{(0)} \\ a_2^{(0)} \\ a_3^{(0)} \\ a_{11}^{(0)} \\ a_{12}^{(0)} \\ a_{13}^{(0)} \\ a_{21}^{(0)} \\ a_{22}^{(0)} \\ a_{23}^{(0)} \\ a_{31}^{(0)} \\ a_{32}^{(0)} \\ a_{33}^{(0)} \\ a^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 8: Linking Round 0 to Round 1

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} s \\ a^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \\ a_{11}^{(1)} \\ a_{12}^{(1)} \\ a_{13}^{(1)} \\ a_{21}^{(1)} \\ a_{22}^{(1)} \\ a_{23}^{(1)} \\ a_{31}^{(1)} \\ a_{32}^{(1)} \\ a_{33}^{(1)} \end{pmatrix} = \begin{pmatrix} \langle s \rangle_{111}^{(1)} \\ \langle s \rangle_{121}^{(1)} \\ \langle s \rangle_{122}^{(1)} \\ \langle s \rangle_{123}^{(1)} \\ \langle s \rangle_{232}^{(1)} \\ \langle s \rangle_{311}^{(1)} \\ \langle s \rangle_{312}^{(1)} \\ \langle s \rangle_{321}^{(1)} \\ \langle s \rangle_{323}^{(1)} \end{pmatrix}$$

Figure 9: What $\mathcal{A}$ learns from Round 1

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} s \\ a^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_{11}^{(2)} \\ a_{12}^{(2)} \\ a_{13}^{(2)} \\ a_{21}^{(2)} \\ a_{22}^{(2)} \\ a_{23}^{(2)} \\ a_{31}^{(2)} \\ a_{32}^{(2)} \\ a_{33}^{(2)} \end{pmatrix} = \begin{pmatrix} \langle s \rangle_{111} \\ \langle s \rangle_{113} \\ \langle s \rangle_{123} \\ \langle s \rangle_{211} \\ \langle s \rangle_{213} \\ \langle s \rangle_{221} \\ \langle s \rangle_{231} \\ \langle s \rangle_{233} \\ \langle s \rangle_{322} \end{pmatrix}$$

Figure 10: What $\mathcal{A}$ learns from Round 2

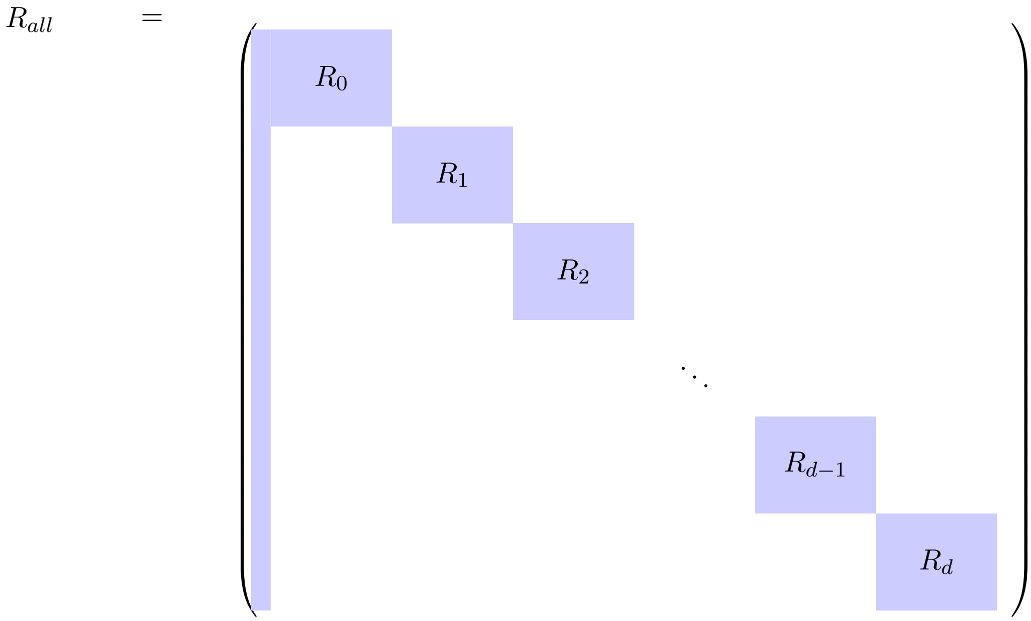


Figure 11: R_{all}

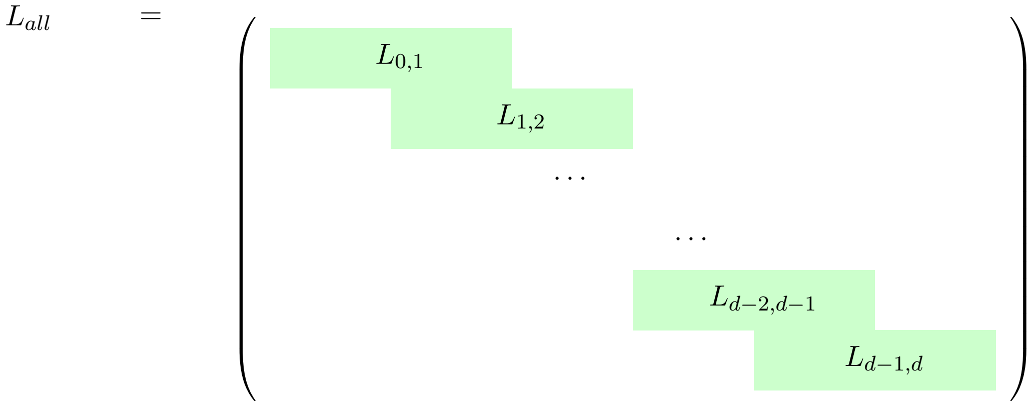


Figure 12: L_{all}

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. pages 101–111, 1998.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sha81] Adi Shamir. The generation of cryptographically strong pseudo-random sequences. page 1, 1981.