Coded DPF Proofs

Albert Gao / sixiangg

03/19/2022

1 Context

We explore the problem of retrieving an item from a server without revealing which item is retrieved. This general class of problems is referred to as private information retrieval (PIR), first introduced in [1]. One class of approaches to this problem makes use of distributed point functions (DPF), first introduced in [2], with improvements in [3] and [4]. Another direction to the problem of PIR concerns the storage model of servers. Instead of assuming multiple servers that replicate the same data set, some recent work such as [5] looks at PIR involving servers with coded storage. We show that the DPF technique can be applied to the coded setting while maintaining polylogarithmic communication complexity.

Before adapting our approach to coded storage, we first discuss PIR and DPF in the setting of replicated storage as well as some of the extensions. As a rough example, suppose each server contains the same $n = 2^m$ items and a client desires item index $\alpha \in \{0,1\}^m$. Suppose each server $j \in [p]$ has access to two functions $h, g_j : \{0,1\}^m \to \mathbb{F}$ such that $\mathbb{F}$ is a finite field, $h(\beta)$ is the data at index β, and $f = \sum_j g_j$ is a coordinate function that vanishes everwhere but α, where $f(\alpha) = 1$. Then as long as each g_j by itself does not reveal α and servers cannot collude, we can compute $h(\alpha) = (h \cdot f)(\alpha) = \sum_\beta h(\beta) \sum_j g_j(\beta) = \sum_j \sum_\beta h(\beta) g_j(\beta)$, where each server j only communicates $\sum_\beta h(\beta) g_j(\beta)$ to the client. We now make precise a more general definition of such sharing of point functions.

Definition 1.1. (Point Function). For $m \in \mathbb{Z}^+, \alpha \in \{0,1\}^m$, the point function $f_\alpha : \{0,1\}^m \to \mathbb{F}$ is defined via

$$f_\alpha(\beta) = \begin{cases} 1, & \beta = \alpha \\ 0, & \beta \neq \alpha \end{cases}.$$

In order to share point functions among p servers, the client generates p relatively short keys so the servers can evaluate each of the shared functions without too much trouble. Here, we assume one-way functions exist, which implies pseudorandom generators (PRGs) exist.

Definition 1.2. (Distributed Point Function: Syntax). A p-party distributed point function (DPF) is a tuple of algorithms (Gen, Eval, Rec) with the following syntax.

- $\mathsf{Gen}(1^\lambda, \alpha)$ is a PPT algorithm that outputs p keys $(k_1, \cdots, k_p)$, where 1^λ is the security parameter and $\alpha \in \{0,1\}^m$.
- $\mathsf{Eval}(k, \beta)$ is a PPT algorithm that outputs some κ in some set A, where $\beta \in \{0,1\}^m$.
- $\mathsf{Rec}(\overrightarrow{\kappa_i})$ is a PPT algorithm that outputs some $\mu \in \mathbb{F}$, where $\overrightarrow{\kappa_i} \in A^p$.

DPF under the setting of at most a unresponsive failures (or b Byzantine servers) is where Eval may instead return empty string (or arbitrary strings) for at most a (or b) servers. We now specify what it means for a DPF to be secure.

Definition 1.3. (Distributed Point Function: Security). A p-party t-secure DPF is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Rec})$ with the following properties.

- **Correctness:** For any nonempty $\alpha \in \{0,1\}^*$, if $(k_1, \cdots, k_p) \leftarrow \mathsf{Gen}(1^\lambda, \alpha)$, then for any $\beta \in \{0,1\}^{|\alpha|}$ we have $\Pr\left[\mathsf{Rec}\left(\overrightarrow{\mathsf{Eval}(k_i, \beta)}\right) = f_\alpha(\beta)\right] = 1$.
- **Secrecy:** For any set $S \subseteq [n]$ of size t, there exists a PPT algorithm Sim such that for every sequence of $(\alpha_\lambda \in \{0,1\}^\lambda)$ where $\lambda \in \mathbb{Z}^+$, the outputs of the following experiments Real and Ideal are computationally indistinguishable.
  - $\mathsf{Real}(1^\lambda) : (k_1, \cdots, k_p) \leftarrow \mathsf{Gen}(1^\lambda, \alpha_\lambda)$. Output $(k_j)_{j \in S}$.
  - $\mathsf{Ideal}(1^\lambda)$: Output $\mathsf{Sim}(1^\lambda)$.

Note that a secure and efficient DPF scheme under this definition does not automatically imply a secure and efficient PIR scheme. We will clarify this point later.

2 Multi-Party DPF

We extend two-party DPF construction from [4] to the p-party setting.

Theorem 2.1. *A p-party 1-secure DPF under the setting of at most $p-2$ unresponsive failures (or, less than $\frac{p}{2}$ Byzantine failures (?)) exists.*

Proof. The tuple of algorithms is as specified in Algorithm 1.

<u>Claim:</u> The algorithms satisfy correctness.

Proof Sketch. Let $\alpha \in \{0,1\}^m$ denote our desired index. Let $\beta \in \{0,1\}^m$ denote the currently evaluated index. Let $(\alpha = \beta)_{\leq i}$ denote $\bigwedge_{1 \leq j \leq i} \alpha_j = \beta_j$ where $(\alpha = \beta)_{\leq 0}$ is always true. Let $s_j^{(i)}$ and $t_{j,k}^{(i)}$ denote seeds and control bits as in algorithm Gen, where $j \in [p]$, $k \in [p^-] = [p] \backslash \{1\}$, $i \in [m] \cup \{0\}$. We first show that the following invariants are maintained for each iteration $i \in [m] \cup \{0\}$.

- Let $[s_{a,b}^{(i)}] = s_a^{(i)} \oplus s_b^{(i)}$ denote the "sharing" of seed between parties a and b at round i. Then $[s_{a,b}^{(i)}] = 0$ if $\neg(\alpha = \beta)_{\leq i}$.
- Let $[t_{k,a,b}^{(i)}] = t_{a,k}^{(i)} \oplus t_{b,k}^{(i)}$ with $a < b$ and $k \in [p^-]$ denote the "sharing" of control bits.
  - If $\neg(\alpha = \beta)_{\leq i}$, then $[t_{k,a,b}^{(i)}] = 0$.
  - If $(\alpha = \beta)_{\leq i}$, then $[t_{k,a,b}^{(i)}] = \begin{cases} 0, & k \neq a \wedge k \neq b \\ 1, & k = a \vee k = b \end{cases}$.

To clarify the above conditions on control bits, for each iteration we may construct the matrix

$$A^{(i)} = \begin{pmatrix} t_{1,2} & t_{1,3} & \cdots & t_{1,p} \\ \boxed{t_{2,2}} & t_{2,3} & \cdots & t_{2,p} \\ t_{3,2} & \boxed{t_{3,3}} & \cdots & t_{3,p} \\ \vdots & \vdots & \ddots & \vdots \\ t_{p,2} & t_{p,3} & \cdots & \boxed{t_{p,p}} \end{pmatrix} \in \mathbb{F}_2^{p \times p-1},$$

Algorithm 1 p-Party Distributed Point Function

Let $G : \{0,1\}^\lambda \to \{0,1\}^{2(\lambda+p-1)}$ be a pseudorandom generator. Notation-wise, if b is a bit, we use $\overline{b}$ to denote $b \oplus 1$. $[p]$ denotes set of integers $\{1, \cdots, p\}$ and $[p^-]$ denotes set of integers $\{2, \cdots, p\} = [p] \setminus \{1\}$.

$\mathsf{Gen}(1^\lambda, \alpha)$:

Let $\alpha_1, \cdots, \alpha_m$ be the bit decomposition of α.
Sample $s_j^{(0)} \leftarrow \{0,1\}^\lambda$ for each $j \in [p]$.
Let $t_{j,k}^{(0)} \leftarrow \begin{cases} 0, & j \neq k \\ 1, & j = k \end{cases}$, for $j \in [p], k \in [p^-]$.
for $i = 1$ to m **do**
  $s_j^0 || t_{j,2}^0 || \cdots || t_{j,p}^0 || s_j^1 || t_{j,2}^1 || \cdots || t_{j,p}^1 \leftarrow G(s_j^{(i-1)})$, for $j \in [p]$.
  $CW_{s_k} \leftarrow s_k^{\overline{\alpha_i}} \oplus s_1^{\overline{\alpha_i}}$, for $k \in [p^-]$.
  $CW_{t_{j,k}^\beta} \leftarrow \begin{cases} t_{1,k}^\beta \oplus t_{j,k}^\beta, & j \neq k \\ t_{1,k}^\beta \oplus t_{j,k}^\beta \oplus \overline{\alpha_i} \oplus \beta, & j = k \end{cases}$, for $j, k \in [p^-], \beta \in \{0,1\}$.
  $CW_k^{(i)} \leftarrow CW_{s_k} || \{CW_{t_{j,k}^0}\}_{j \in [p^-]} || \{CW_{t_{j,k}^1}\}_{j \in [p^-]}$, for $k \in [p^-]$.
  $CW^{(i)} \leftarrow CW_2^{(i)} || \cdots || CW_p^{(i)}$.
  $s_j^{(i)} \leftarrow s_j^{\alpha_i} \bigoplus_{k \in [p^-]} t_{j,k}^{(i-1)} \cdot CW_{s_k}$, for $j \in [p]$
  $t_{j,l}^{(i)} \leftarrow t_{j,l}^{\alpha_i} \bigoplus_{k \in [p^-]} t_{j,k}^{(i-1)} \cdot CW_{t_{j,k}^{\alpha_i}}$, for $j \in [p], l \in [p^-]$.
end for
$CW^{(m+1)} \leftarrow 2 \oplus s_1^{(m)} \oplus s_2^{(m)} || \cdots || p \oplus s_1^{(m)} \oplus s_p^{(m)}$, where $2, \cdots, p$ expressed in binary.
$k_j \leftarrow s_j^{(0)} || \{t_{j,k}^{(0)}\}_{k \in [p^-]} || CW^{(1)} || \cdots || CW^{(m+1)}$.
return $(k_1, \cdots, k_p)$.

$\mathsf{Eval}(k_j, \beta)$:

Parse $k_j = s^{(0)} || \{t_k^{(0)}\}_{k \in [p^-]} || CW^{(1)} || \cdots || CW^{(m+1)}$.
for $i = 1$ to m **do**
  Parse $CW^{(i)} = \{CW_k^{(i)}\}_{k \in [p^-]}$ and $CW_k^{(i)} = CW_{s_k} || \{CW_{t_{j,k}^0}\}_{j \in [p^-]} || \{CW_{t_{j,k}^1}\}_{j \in [p^-]}$.
  $\tau^{(i)} \leftarrow G(s^{(i-1)}) \bigoplus_{k \in [p^-]} \left(t_k^{(i-1)} \cdot \left(CW_{s_k} || \{CW_{t_{j,k}^0}\}_{j \in [p^-]} || CW_{s_k} || \{CW_{t_{j,k}^1}\}_{j \in [p^-]} \right) \right)$.
  Parse $\tau^{(i)} = s^0 || \{t_k^0\}_{k \in [p^-]} || s^1 || \{t_k^1\}_{k \in [p^-]}$.
  $s^{(i)} \leftarrow s^{\beta_i}, t_k^{(i)} \leftarrow t_k^{\beta_i}$, for $k \in [p^-]$.
end for
Parse $CW^{(m+1)} = CW_2^{(m+1)} || \cdots || CW_p^{(m+1)}$.
return $s^{(m)} \bigoplus_{k \in [p^-]} \left(t_k^{(m)} \cdot CW_k^{(m+1)} \right)$.

$\mathsf{Rec}(s_1, \cdots, s_p)$:

$a := 0, b := 0$.
for $j, k \in [p]$ with $j < k$ **do**
  if $s_j \neq s_k$ **then**
    $a := a + 1$.
  else
    $b := b + 1$.
  end if
end for
if $a > b$ **then**
  return $1_{\mathbb{F}}$.
else
  return $0_{\mathbb{F}}$.
end if

where superscript (i) is omitted on each entry. If we consider only one column $\{t_{j,k}\}$ of A then we desire the property that $t_{1,k} = t_{2,k} = \cdots = t_{k-1,k} = t_{k+1,k} = \cdots = t_{p,k} \neq t_{k,k}$, as long as $(\alpha = \beta)_{\leq i}$. In words, we want the boxed element to differ from all other elements in the column.

We show the conditions hold by induction on i.

- $i = 0$. This is clear.
- $i > 0$. Assume the conditions hold for $i - 1$. Let $[s^c_{a,b}] = s^c_a \oplus s^c_b$ with $a, b \in [p^-]$, $c \in \{0, 1\}$ and s^c_a and s^c_b are parts of the output from the pseudorandom generator from iteration i as in algorithm description. Similarly let $[t^c_{k,a,b}] = t^c_{a,k} \oplus t^c_{b,k}$. Let $CW_{s_k}, CW_{t^c_{j,k}}$ denote correction words in this iteration. We have two cases.
  - $\neg(\alpha = \beta)_{\leq i-1}$. Then each column of $A^{(i-1)}$ has identical elements. Additionally, all $s^{(i-1)}_{a,b}$ identical, which implies generated seeds and control bits from pseudorandom generator are identical among all parties. It follows that $[s^{(i)}_{a,b}] = [s^{\beta_i}_{a,b}] \bigoplus_{k \in [p^-]} [t^{(i-1)}_{k,a,b}] \cdot CW_{s_k} = 0 \bigoplus_k 0 = 0$. The same holds for $[t^{(i)}_{k,a,b}]$.
  - $(\alpha = \beta)_{\leq i-1}$. Then we consider two cases.
    * $a = 1$. Let $b \in [p^-]$. Then

$$\begin{aligned}
[s^{(i)}_{a,b}] &= [s^{\beta_i}_{a,b}] \bigoplus_{k \in [p^-]} [t^{(i-1)}_{k,a,b}] \cdot [s^{\overline{\alpha_i}}_{1,k}] \\
&= [s^{\beta_i}_{a,b}] \oplus [t^{(i-1)}_{b,1,b}] \cdot [s^{\overline{\alpha_i}}_{1,b}] \\
&= [s^{\beta_i}_{1,b}] \oplus [s^{\overline{\alpha_i}}_{1,b}]
\end{aligned}$$

      which is 0 if $\neg(\alpha = \beta)_{\leq i} \Rightarrow \alpha_i \neq \beta_i$. Further,

$$\begin{aligned}
[t^{(i)}_{k,a,b}] &= [t^{\beta_i}_{k,a,b}] \bigoplus_{j \in [p^-]} [t^{(i-1)}_{j,a,b}] \cdot [t^{\beta_i}_{k,1,j}] \oplus \mathbf{1}_{k=j}(\overline{\alpha_i} \oplus \beta_i) \\
&= [t^{\beta_i}_{k,a,b}] \oplus [t^{(i-1)}_{b,a,b}] \cdot [t^{\beta_i}_{k,1,b}] \oplus \mathbf{1}_{k=b}(\overline{\alpha_i} \oplus \beta_i) \\
&= 0 \oplus \mathbf{1}_{k=b}(\overline{\alpha_i} \oplus \beta_i) \\
&= \begin{cases} 0, & k \neq b \\ 0, & k = b, \neg(\alpha = \beta)_{\leq i} \\ 1, & k = b, (\alpha = \beta)_{\leq i} \end{cases} .
\end{aligned}$$

    * $a \neq 1$. Let $b \in [p^-]$ such that $a < b$. **Exercise:** Show if $\neg(\alpha = \beta)_{\leq i}$ then $[s^{(i)}_{a,b}] = 0$. Finally, we have

$$\begin{aligned}
[t^{(i)}_{k,a,b}] &= [t^{\beta_i}_{k,a,b}] \bigoplus_{j \in [p^-]} [t^{(i-1)}_{j,a,b}] \cdot [t^{\beta_i}_{k,1,j}] \\
&= [t^{\beta_i}_{k,a,b}] \oplus [t^{(i-1)}_{a,a,b}] \cdot [t^{\beta_i}_{k,1,a}] \oplus \mathbf{1}_{k=a}(\overline{\alpha_i} \oplus \beta_i) \\
&\oplus [t^{(i-1)}_{b,a,b}] \cdot [t^{\beta_i}_{k,1,b}] \oplus \mathbf{1}_{k=b}(\overline{\alpha_i} \oplus \beta_i) \\
&= \begin{cases} 0, & k \neq a \wedge k \neq b \\ 0, & (k = a \vee k = b) \wedge \alpha_i \neq \beta_i \\ 1, & (k = a \vee k = b) \wedge \alpha_i = \beta_i \end{cases} .
\end{aligned}$$

Hence by induction the conditions hold for all i. **Exercise:** Show that this implies correctness of Rec. □

Claim: The algorithms satisfy secrecy.

Proof. Todo. □

□

Theorem 2.2. *A p-party t-secure DPF under the setting of at most $p-t-1$ unresponsive failures (or, less than $\frac{p-t-1}{2}$ Byzantine failures (?)) exists, for any $1 \le t \le p-1$.*

Proof. The tuple of algorithms is as specified in Algorithm 2.

Algorithm 2 p-Party Collusion-Tolerating Distributed Point Function

Notation-wise, if b is a bit, we use $\bar{b}$ to denote $b \oplus 1$. $[p]$ denotes set of integers $\{1, \cdots, p\}$ and $[p^-]$ denotes set of integers $\{2, \cdots, p\} = [p] \setminus \{1\}$.

$\mathsf{Gen}(1^\lambda, \alpha)$:

```
Let α_1, ···, α_m be the bit decomposition of α.
Sample a_1, ···, a_{m+1} ← F*.
Sample degree t polynomials q_1, ···, q_{m+1} where q_i(0) = a_i.
Let x_1, ···, x_p ∈ F* denote distinct elements.
Let [a_i] denote sharing of a_i, where each party j ∈ [p] should have access to q_i(x_j).
for i = 1 to m do
    Sample c_{j,0}, c_{j,1} ← F* for i < j ≤ m + 1.
    Find {d_k}_{i≤k≤m} such that a_i d_i + a_{i+1} c_{i+1, \overline{α_i}} = 0_F.
    Form CW^{(i)} = {c_{j,0}||c_{j,1}}_{i<j≤m+1}||{d_k}_{i≤k≤m}.
    For each i < j ≤ m + 1, update a_j ← a_j · c_{j,β_i} + a_{j−1} · d_{j−1}.
end for
CW^{(m+1)} ← (a_{m+1})^{−1}.
{k_j} ← [a_1]|| ··· ||[a_{m+1}]||CW^{(1)}|| ··· ||CW^{(m+1)}.
return (k_1, ···, k_p).
```

$\mathsf{Eval}(k_j, \beta)$:

```
Parse k_j = b_1|| ··· ||b_{m+1}||CW^{(1)}|| ··· ||CW^{(m+1)}.
for i = 1 to m do
    Parse CW^{(i)} = {c_{j,0}||c_{j,1}}_{i<j≤m+1}||{d_k}_{i≤k≤m}.
    For each i < j ≤ m + 1, update b_j ← b_j · c_{j,β_i} + b_{j−1} · d_{j−1}.
end for
Set b_{m+1} ← b_{m+1} · CW^{(m+1)}.
return b_{m+1}.
```

$\mathsf{Rec}(s_1, \cdots, s_p)$:

```
return Constant coefficient of polynomial interpolation of \vec{s_j}.
```

□

References

[1] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995.

[2] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 640–658. Springer, 2014.

[3] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer, 2015.

[4] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. Cryptology ePrint Archive, Report 2018/707, 2018.

[5] Razane Tajeddine, Oliver W Gnilke, David Karpuk, Ragnar Freij-Hollanti, and Camilla Hollanti. Private information retrieval from coded storage systems with colluding, byzantine, and unresponsive servers. *IEEE Transactions on information theory*, 65(6):3898–3906, 2019.