The Name of the Title Is Hope

tbd

ABSTRACT

tbd.

CCS CONCEPTS

• **Theory of computation → Cryptographic primitives**.

KEYWORDS

tbd

ACM Reference Format:
tbd. tbd. The Name of the Title Is Hope. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/tbd

1 INTRODUCTION

tbd

2 PRELIMINARY

2.1 Basic Notations

Point and multi-point functions. Given a domain size N and Abelian group $\mathbb{G}$, a *point function* $f_{\alpha,\beta} : [N] \to \mathbb{G}$ for $\alpha \in [N]$ and $\beta \in \mathbb{G}$ evaluates to β on input α and to $0 \in \mathbb{G}$ on all other inputs. We denote by $\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$ the representation of such a point function. A *t-point function* $f_{A,B} : [N] \to \mathbb{G}$ for $A = (\alpha_1, \cdots \alpha_t) \in [N]^t$ and $B = (\beta_1, \cdots, \beta_t) \in \mathbb{G}^t$ evaluates to β_i on input α_i for $1 \le i \le t$ and to 0 on all other inputs. Denote $\hat{f}_{A,B}(N, \hat{\mathbb{G}}, t, A, B)$ the representation of such a t-point function. Call the collection of all t-point functions for all t *multi-point functions*.

Enote: MPF. Also representation of groups.

2.2 Distributed Multi-Point Functions

Enote: should directly adapt to multi-point function case

We begin by defining a slightly generalized notion of distributed point functions (DPFs), which accounts for the extra parameter $\mathbb{G}'$. Yaxin: What is $\mathbb{G}'$?

Definition 1 (DPF [1, 3]). *A (2-party)* distributed point function (DPF) *is a triple of algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Eval}_0, \mathsf{Eval}_1)$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, \hat{f}_{\alpha,\beta}) \to (k_0, k_1)$*: On input security parameter* $\lambda \in \mathbb{N}$ *and point function description* $\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$*, the (randomized) key generation algorithm* Gen *returns a pair of keys* $k_0, k_1 \in \{0,1\}^*$*. We assume that* N *and* $\mathbb{G}$ *are determined by each key.*
- $\mathsf{Eval}_i(k_i, x) \to y_i$*: On input key* $k_i \in \{0,1\}^*$ *and input* $x \in [N]$ *the (deterministic) evaluation algorithm of server* i*,* Eval_i *returns* $y_i \in \mathbb{G}$*.*

We require Π *to satisfy the following requirements:*

- ***Correctness:*** *For every* λ*,* $\hat{f} = \hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$ *such that* $\beta \in \mathbb{G}$*, and* $x \in [N]$*, if* $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f})$*, then*

$$\Pr\left[\sum_{i=0}^{1} \mathsf{Eval}_i(k_i, x) = f_{\alpha,\beta}(x)\right] = 1$$

- ***Security:*** *Consider the following semantic security challenge experiment for corrupted server* $i \in \{0,1\}$*:*

(1) *The adversary produces two point function descriptions* $(\hat{f}^0 = (N, \hat{\mathbb{G}}, \alpha_0, \beta_0), \hat{f}^1 = (N, \hat{\mathbb{G}}, \alpha_1, \beta_1)) \leftarrow \mathcal{A}(1^\lambda)$*, where* $\alpha_i \in [N]$ *and* $\beta_i \in \mathbb{G}$*.*
(2) *The challenger samples* $b \leftarrow \{0,1\}$ *and* $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}^b)$*.*
(3) *The adversary outputs a guess* $b' \leftarrow \mathcal{A}(k_i)$*.*

Denote by $\mathsf{Adv}(1^\lambda, \mathcal{A}, i) = \Pr[b = b'] - 1/2$ *the advantage of* $\mathcal{A}$ *in guessing* b *in the above experiment. For every non-uniform polynomial time adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\mathsf{Adv}(1^\lambda, \mathcal{A}, i) \le \nu(\lambda)$ *for all* $\lambda \in \mathbb{N}$*.*

Definition 2 (DMPF). *A (2-party)* distributed multi-point function (DMPF) *is a triple of algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Eval}_0, \mathsf{Eval}_1)$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, \hat{f}_{A,B}) \to (k_0, k_1)$*: On input security parameter* $\lambda \in \mathbb{N}$ *and point function description* $\hat{f}_{A,B} = (N, \hat{\mathbb{G}}, t, A, B)$*, the (randomized) key generation algorithm* Gen *returns a pair of keys* $k_0, k_1 \in \{0,1\}^*$*.*
- $\mathsf{Eval}_i(k_i, x) \to y_i$*: On input key* $k_i \in \{0,1\}^*$ *and input* $x \in [N]$ *the (deterministic) evaluation algorithm of server* i*,* Eval_i *returns* $y_i \in \mathbb{G}$*.*

We require Π *to satisfy the following requirements:*

- ***Correctness:*** *For every* λ*,* $\hat{f} = \hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$ *such that* $\beta \in \mathbb{G}$*, and* $x \in [N]$*, if* $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f})$*, then*

$$\Pr\left[\sum_{i=0}^{1} \mathsf{Eval}_i(k_i, x) = f_{\alpha,\beta}(x)\right] = 1$$

- ***Security:*** *Consider the following semantic security challenge experiment for corrupted server* $i \in \{0,1\}$*:*

(1) *The adversary produces two* t*-point function descriptions* $(\hat{f}^0 = (N, \hat{\mathbb{G}}, t, A_0, B_0), \hat{f}^1 = (N, \hat{\mathbb{G}}, t, A_1, B_1)) \leftarrow \mathcal{A}(1^\lambda)$*, where* $\alpha_i \in [N]$ *and* $\beta_i \in \mathbb{G}$*.*
(2) *The challenger samples* $b \leftarrow \{0,1\}$ *and* $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}^b)$*.*
(3) *The adversary outputs a guess* $b' \leftarrow \mathcal{A}(k_i)$*.*

Denote by $\mathsf{Adv}(1^\lambda, \mathcal{A}, i) = \Pr[b = b'] - 1/2$ *the advantage of* $\mathcal{A}$ *in guessing* b *in the above experiment. For every non-uniform polynomial time adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\mathsf{Adv}(1^\lambda, \mathcal{A}, i) \le \nu(\lambda)$ *for all* $\lambda \in \mathbb{N}$*.*

Conference acronym 'XX, tbd, tbd

ACM ISBN tbd…$15.00
https://doi.org/tbd

We will also be interested in applying the evaluation algorithm on *all* inputs. Given a DMPF $(\mathsf{Gen}, \mathsf{Eval}_0, \mathsf{Eval}_1)$, we denote by $\mathsf{FullEval}_i$ an algorithm which computes Eval_i on every input x. Hence, $\mathsf{FullEval}_i$ receives only a key k_i as input.

2.3 Batch Codes

combinatorial/probabilistic batch codes, with cuckoo hashing a concrete instantiation

2.4 Oblivious Key-Value Stores

DEFINITION 3 (OKVS[2, 4]). *An Oblivious Key-Value Stores (OKVS) scheme is a pair of randomized algorithms* $(\mathsf{Encode}_r, \mathsf{Decode}_r)$ *with respect to a statistical security parameter* λ_{stat} *and a computational security parameter* λ*, a randomness space* $\{0,1\}^\kappa$*, a key space* $\mathcal{K}$*, a value space* $\mathcal{V}$*, input length n and output length* $m(n)$*. The algorithms are of the following syntax:*

- $\mathsf{Encode}_r(\{(k_1,v_1),(k_2,v_2),\cdots,(k_n,v_n)\}) \to P$*: On input n key-value pairs with distinct keys, the encode algorithm with randomness r in the randomness space outputs an encoding* $P \in \mathcal{V}^m \cup \perp$.
- $\mathsf{Decode}_r(P,k) \to v$*: On input an encoding from* $\mathcal{V}^m$ *and a key* $k \in \mathcal{K}$*, output a value* v.

We require the scheme to satisfy

- ***Correctness:*** *For every* $S \in (\mathcal{K}\times\mathcal{V})^n$, $\Pr_{r\leftarrow\{0,1\}^\kappa}[\mathsf{Encode}_r(S) = \perp] \le 2^{-\lambda_{\mathsf{stat}}}$.
- ***Obliviousness:*** *Given any distinct key sets* $\{k_1^0,k_2^0,\cdots,k_n^0\}$ *and* $\{k_1^1,k_2^1,\cdots,k_n^1\}$ *that are different, if they are paired with random values then their encodings are computationally indistinguishable, i.e.,*

$$\{r, \mathsf{Encode}_r(\{(k_1^0,v_1),\cdots,(k_n^0,v_n)\})\}_{v_1,\cdots,v_n\leftarrow\mathcal{V},r\leftarrow\{0,1\}^\kappa}$$
$$\approx_c\{r, \mathsf{Encode}_r(\{(k_1^1,v_1),\cdots,(k_n^1,v_n)\})\}_{v_1,\cdots,v_n\leftarrow\mathcal{V},r\leftarrow\{0,1\}^\kappa}$$

One can obtain a linear OKVS *if in addition require:*

- ***Linearity:*** *There exists a function family* $\{\mathsf{row}_r : \mathcal{K} \to \mathcal{V}^m\}_{r\in\{0,1\}^\kappa}$ *such that* $\mathsf{Decode}_r(P,k) = \langle \mathsf{row}_r(k), P\rangle$.

The Encode process for a linear OKVS is the process of sampling a random P from the set of solutions of the linear system $\{\langle \mathsf{row}_r(k_i), P\rangle = v_i\}_{1\le i\le n}$.

We evaluate an OKVS scheme by its encoding size (output length m), encoding time and decoding time. We stress the following two (linear) OKVS constructions:

CONSTRUCTION 1 (POLYNOMIAL). *Suppose* $\mathcal{K} = \mathcal{V} = \mathbb{F}$ *is a field. Set*

- $\mathsf{Encode}(\{(k_i,v_i)\}_{1\le i\le n}) \to P$ *where P is the coeffients of a* $(n-1)$*-degree* $\mathbb{F}$*-polynomial* g_P *that* $g_P(k_i) = v_i$ *for* $1 \le i \le n$.
- $\mathsf{Decode}(P,k) \to g_P(k)$.

The polynomial OKVS possesses an optimal encoding size $m = n$, but the Encode process is a polynomial interpolation which is only known to be achieved in time $O(n\log^2 n)$. The time for a single decoding is $O(n)$ and that for batched decodings is (amortized) $O(\log^2 n)$.

An alternative construction that has near optimal encoding size but much better running time is as follows.

CONSTRUCTION 2 (3-HASH GARBLED CUCKOO TABLE (3H-GCT)[2, 4]). *Suppose* $\mathcal{V} = \mathbb{F}$ *is a field. Set* $\mathsf{row}_r(k) := \mathsf{row}_r^{\mathsf{sparse}}(k)||\mathsf{row}_r^{\mathsf{dense}}(k)$ *where* $\mathsf{row}_r^{\mathsf{sparse}}$ *outputs a uniformly random weight-w vector in* $\{0,1\}^{m_1}$*, and* $\mathsf{row}_r^{\mathsf{dense}}(k)$ *outputs a short dense vector in* $\mathbb{F}^{m_2}$.

- $\mathsf{Encode}(\{(k_i,v_i)\}_{1\le i\le n}) \to P$ *where P is solved from the system* $\{\langle \mathsf{row}_r(k_i), P\rangle = v_i\}_{1\le i\le n}$ *using the triangulation algorithm in [4].*
- $\mathsf{Decode}(P,k) \to \langle \mathsf{row}_r(k), P\rangle$.

This OKVS construction features a linear encoding time, constant decoding time while having a linear encoding size.

TBD: Carefully(!) recompute the comparison table for OKVS and insert

We take $w = 3$, the most common option that outruns other choices of w in terms of running time. Restating the conclusion in [4]: given n and λ_{stat}, the choices of e and $\hat{g}$ are $e = 1.223 + \frac{\lambda_{\mathsf{stat}}+9.2}{4.144n^{0.55}}$ and $\hat{g} = \frac{\lambda_{\mathsf{stat}}}{\log_2(en)}$.

TBD: mention some connections to cuckoo hashing

3 NEW DMPF CONSTRUCTIONS

In this section, we display two new constructions of DMPF that follow the same construction paradigm shown in fig. 1.

We begin by introducing the DMPF paradigm in fig. 1, which is based on the idea of the DPF construction in [1]. Each key $k_b (b = 0,1)$ generated by $\mathsf{Gen}(\hat{f}_{A,B})$ can span a height-n (n is the input length of $\hat{f}_{A,B}$) complete binary tree (call it the evaluation tree), with which party b can evaluate the input $x = x_1\cdots x_n$ by starting from the root of this tree, on the ith layer going left if $x_i = 0$ and going right if $x_i = 1$, until reaching a leaf node then computing the result according to this leaf node. Each node of this tree is associated with a λ-bit seed and a l-bit sign. For a node on the ith layer with seed sd and sign sig, its children's seeds and signs are generated by $\mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{Correction}$, where the Correction is determined by the parent node, its sign sig and a correction word $CW^{(i)}$ associated with that layer.

Correct and ConvCorrect are linear in the second input argument sign, i.e., for all $\bar{x}$ and CW, $\mathsf{Correct}(\bar{x}, 0^l, CW) = 0^{\lambda+2l}$ and $\mathsf{Correct}(\bar{x}, \mathsf{sign}, CW) \oplus \mathsf{Correct}(\bar{x}, \mathsf{sign}', CW) = \mathsf{Correct}(\bar{x}, \mathsf{sign} \oplus \mathsf{sign}', CW)$. The same should hold for ConvCorrect.

3.1 Big-State DMPF

TBD: explain

Public parameters:
The t-point function family $\{f_{A,B}\}$ with t an upperbound of the number of nonzero points, input domain $[N] = \{0,1\}^n$ and the output group $\mathbb{G}$.
Suppose there is a public PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda+2l}$. Parse $G = G_0||G_1$ to the left half and right half.
Suppose there is a public PRG $G_{\text{convert}} : \{0,1\}^\lambda \to \mathbb{G}$.

procedure $\text{Gen}(1^\lambda, \hat{f}_{A,B})$
Denote $A = (\alpha_1, \cdots, \alpha_t)$ in lexicographical order, $B = (\beta_1, \cdots, \beta_t)$. If $|A| < t$, extend A to size-t with arbitrary $\{0,1\}^n$ strings and B with 0's.
For $0 \le i \le n-1$, let $A^{(i)}$ denote the sorted and deduplicated list of i-bit prefixes of strings in A. Specifically, $A^{(0)} = [\epsilon]$.
For $0 \le i \le n-1$ and $b = 0, 1$, initialize empty lists $\mathsf{seed}_b^{(i)}$ and $\mathsf{sign}_b^{(i)}$.
$\mathsf{Initialize}(\{\mathsf{seed}_b^{(0)}, \mathsf{sign}_b^{(0)}\}_{b=0,1})$.
for $i = 1$ to n **do**
$CW^{(i)} \leftarrow \mathsf{GenCW}(i, A, \{\mathsf{seed}_b^{(i-1)}, \mathsf{sign}_b^{(i-1)}\}_{b=0,1})$.
for $k = 1$ to $|A^{(i-1)}|$ and $z = 0, 1$ **do**
Compute $C_{\mathsf{seed},b}||C_{\mathsf{sign}^0,b}||C_{\mathsf{sign}^1,b} \leftarrow \mathsf{Correct}(A^{(i-1)}[k], \mathsf{sign}_b^{(i-1)}[k], CW^{(i)})$ for $b = 0, 1$.
if $A^{(i-1)}[k]||z \in A^{(i)}$ **then**
Append the first λ bit of $G_z(\mathsf{seed}_b^{(i-1)}[k]) \oplus (C_{\mathsf{seed},b}||C_{\mathsf{sign}^z,b})$ to $\mathsf{seed}_b^{(i)}$ and the rest to $\mathsf{sign}_b^{(i)}$.
end if
end for
end for
$CW^{(n+1)} \leftarrow \mathsf{GenConvCW}(A, B, \{\mathsf{seed}_b^{(n)}, \mathsf{sign}_b^{(n)}\}_{b=0,1})$.
Set $k_b \leftarrow (\mathsf{seed}_b^{(0)}, \mathsf{sign}_b^{(0)}, CW^{(1)}, CW^{(2)}, \cdots, CW^{(n+1)})$.
return (k_0, k_1).
end procedure

procedure $\text{Eval}_b(1^\lambda, k_b, x)$
Parse $k_b = ([\mathsf{seed}], [\mathsf{sign}], CW^{(1)}, CW^{(2)}, \cdots, CW^{(n+1)})$.
Denote $x = x_1 x_2 \cdots x_n$.
for $i = 1$ to n **do**
$C_{\mathsf{seed}}||C_{\mathsf{sign}^0}||C_{\mathsf{sign}^1} \leftarrow \mathsf{Correct}(x_1 \cdots x_{i-1}, \mathsf{sign}, CW^{(i)})$.
$\mathsf{seed}||\mathsf{sign} \leftarrow (\mathsf{seed} \oplus C_{\mathsf{seed}})||(\mathsf{sign} \oplus C_{\mathsf{sign}^{x_i}})$.
end for
return $(-1)^b \cdot \big(G_{\text{convert}}(\mathsf{seed}) + \mathsf{ConvCorrect}(x, \mathsf{sign}, CW^{(n+1)})\big)$.
end procedure

procedure $\text{FullEval}_b(1^\lambda, k_b)$
Parse $k_b = (\mathsf{seed}^{(0)}, \mathsf{sign}^{(0)}, CW^{(1)}, CW^{(2)}, \cdots, CW^{(n+1)})$.
For $1 \le i \le n$, $\mathsf{Path}^{(i)} \leftarrow$ the lexicographical ordered list of $\{0,1\}^i$. $\mathsf{Path}^{(0)} \leftarrow [\epsilon]$.
for $i = 1$ to n **do**
for $k = 1$ to 2^{i-1} **do**
$C_{\mathsf{seed}}||C_{\mathsf{sign}^0}||C_{\mathsf{sign}^1} \leftarrow \mathsf{Correct}(\mathsf{Path}(i-1)[k], \mathsf{sign}^{(i-1)}[k], CW^{(i)})$.
$\mathsf{seed}^{(i)}[2k]||\mathsf{sign}^{(i)}[2k] \leftarrow G_0(\mathsf{seed}^{(i-1)}[k]) \oplus (C_{\mathsf{seed}}||C_{\mathsf{sign}^0})$.
$\mathsf{seed}^{(i)}[2k+1]||\mathsf{sign}^{(i)}[2k+1] \leftarrow G_1(\mathsf{seed}^{(i-1)}[k]) \oplus (C_{\mathsf{seed}}||C_{\mathsf{sign}^1})$.
end for
end for
for $k = 1$ to 2^n **do**
$\mathsf{Output}[k] \leftarrow (-1)^b \cdot \big(G_{\text{convert}}(\mathsf{seed}^{(n)}[k]) + \mathsf{ConvCorrect}(\mathsf{Path}[k], \mathsf{sign}^{(n)}[k], CW^{(n+1)})\big)$.
end for
return Output.
end procedure

Figure 1: The paradigm of our DMPF schemes. We leave the PRG expand length l, methods Initialize, GenCW, GenConvCW, Correct, ConvCorrect **to be determined by specific constructions.**

Set $l \leftarrow t$, the upperbound of $|A|$.

procedure INITIALIZE($\{\mathsf{seed}_b^{(0)}, \mathsf{sign}_b^{(0)}\}_{b=0,1}$)

  For $b = 0, 1$, let $\mathsf{seed}_b^{(0)} = [r_b]$ where $r_b \xleftarrow{\$} \{0,1\}^\lambda$.

  For $b = 0, 1$, set $\mathsf{sign}_b^{(0)} = [b||0^{t-1}]$.

end procedure

procedure GENCW($i, A, \{\mathsf{seed}_b^{(i-1)}, \mathsf{sign}_b^{(i-1)}\}_{b=0,1}$)

  Let $\{A^{(i)}\}_{0 \le i \le n}$ be defined as in fig. 1.

  Sample a list CW of t random strings from $\{0,1\}^{\lambda+2t}$.

  for $k = 1$ to $|A^{(i-1)}|$ **do**

    Parse $G(\mathsf{seed}_b^{(i-1)}[k]) = \mathsf{seed}_b^0||\mathsf{sign}_b^0||\mathsf{seed}_b^1||\mathsf{sign}_b^1$, for $b = 0, 1$, $\mathsf{seed}_b^0, \mathsf{seed}_b^1 \in \{0,1\}^\lambda$ and $\mathsf{sign}_b^0, \mathsf{sign}_b^1 \in \{0,1\}^t$.

    Compute $\Delta\mathsf{seed}^c = \mathsf{seed}_0^c \oplus \mathsf{seed}_1^c$ and $\Delta\mathsf{sign}^c = \mathsf{sign}_0^c \oplus \mathsf{sign}_1^c$ for $c = 0, 1$.

    Denote path $\leftarrow A^{(i-1)}[k]$.

    if both path$||z$ for $z = 0, 1$ are in $A^{(i)}$ **then**

      $d \leftarrow$ the index of path$||0$ in $A^{(i)}$.

      $CW[d] \leftarrow r||\Delta\mathsf{sign}^0 \oplus e_d||\Delta\mathsf{sign}^1 \oplus e_{d+1}$ where $r \xleftarrow{\$} \{0,1\}^\lambda$, $e_d = 0^{d-1}10^{t-d}$.

    else

      Let z be such that path$||z \in A^{(i)}$.

      $d \leftarrow$ the index of path$||z$ in $A^{(i)}$.

$$CW[d] \leftarrow \begin{cases} \Delta\mathsf{seed}^1||\Delta\mathsf{sign}^0 \oplus e_d||\Delta\mathsf{sign}^1 & z = 0 \\ \Delta\mathsf{seed}^0||\Delta\mathsf{sign}^0||\Delta\mathsf{sign}^1 \oplus e_d & z = 1 \end{cases}.$$

    end if

  end for

  return CW.

end procedure

procedure GENCONVCW($A, B, \{\mathsf{seed}_b^{(n)}, \mathsf{sign}_b^{(n)}\}$)

  Sample a list CW of t random $\mathbb{G}$-elements.

  for $k = 1$ to $|A|$ **do**

    $\Delta g \leftarrow G_{\mathsf{convert}}(\mathsf{seed}_0^{(n)}[k]) - G_{\mathsf{convert}}(\mathsf{seed}_1^{(n)}[k])$.

    $CW[k] \leftarrow (-1)^{\mathsf{sign}_0^{(n)}[k][k]}(\Delta g - B[k])$.

  end for

  return CW.

end procedure

procedure CORRECT($\bar{x}, \mathsf{sign}, CW$)

  return $C_{\mathsf{seed}}||C_{\mathsf{sign}^0}||C_{\mathsf{sign}^1} \leftarrow \sum_{i=1}^t \mathsf{sign}[i] \cdot CW[i]$, where C_{sign^0} and C_{sign^1} are t-bit.

end procedure

procedure CONVCORRECT(x, sign, CW)

  return $\sum_{i=1}^t \mathsf{sign}[i] \cdot CW[i]$.

end procedure

Figure 2: The parameter l and methods' setting that turns the paradigm of DMPF in fig. 1 into the big-state DMPF.

3.2 Batch-Code DMPF

display the batch-code DMPF

3.3 OKVS-based DMPF

TBD: explain

3.4 Comparison

Comparison table dependent to PRG & F-MUL(list the formulas?)
analyze tradeoff
distributed gen advantage

3.5 Distributed Key Generation

4 APPLICATIONS

4.1 PCG for OLE from Ring-LPN

Characterize parameters
show nonregular optimization
plug in new DMPF and show overall optimization

4.2 PSI-WCA

plug in new DMPF and analyze advantage interval
plug in distributed gen

4.3 Heavy-hitters

private heavy-hitter
or parallel ORAM?

5 ACKNOWLEDGMENTS

tbd

REFERENCES

[1] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2018. Function Secret Sharing: Improvements and Extensions. Cryptology ePrint Archive, Paper 2018/707. https://eprint.iacr.org/2018/707 https://eprint.iacr.org/2018/707.

[2] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021. Oblivious Key-Value Stores and Amplification for Private Set Intersection. Cryptology ePrint Archive, Paper 2021/883. https://eprint.iacr.org/2021/883 https://eprint.iacr.org/2021/883.

[3] Niv Gilboa and Yuval Ishai. 2014. Distributed Point Functions and Their Applications. In *Advances in Cryptology – EUROCRYPT 2014*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 640–658.

[4] Srinivasan Raghuraman and Peter Rindal. 2022. Blazing Fast PSI from Improved OKVS and Subfield VOLE. Cryptology ePrint Archive, Paper 2022/320. https://eprint.iacr.org/2022/320 https://eprint.iacr.org/2022/320.

A BATCH-CODE DMPF SCHEME

B SECURITY PROOFS

Set $l \leftarrow 1$.
For $1 \leq i \leq n$, let OKVS_i be an OKVS scheme (definition 3) with key space $\mathcal{K} = \{0,1\}^{i-1}$, value space $\mathcal{V} = \{0,1\}^{\lambda+2}$ and input length t.
let $\mathsf{OKVS}_{\mathsf{convert}}$ be an OKVS scheme with key space $\mathcal{K} = \{0,1\}^n$, value space $\mathcal{V} = \mathbb{G}$ and input length t.

procedure Initialize($\{\mathsf{seed}_b^{(0)}, \mathsf{sign}_b^{(0)}\}_{b=0,1}$)
  For $b = 0,1$, let $\mathsf{seed}_b^{(0)} = [r_b \xleftarrow{\$} \{0,1\}^\lambda]$ and $\mathsf{sign}_b^{(0)} = [b]$.
end procedure

procedure GenCW($i, A, \{\mathsf{seed}_b^{(i-1)}, \mathsf{sign}_b^{(i-1)}\}_{b=0,1}$)
  Let $\{A^{(i)}\}_{0 \leq i \leq n}$ be defined as in fig. 1.
  Sample a list V of t random strings from $\{0,1\}^{\lambda+2}$.
  for $k = 1$ to $|A^{(i-1)}|$ **do**
    Parse $G(\mathsf{seed}_b^{(i-1)}[k]) = \mathsf{seed}_b^0 || \mathsf{sign}_b^0 || \mathsf{seed}_b^1 || \mathsf{sign}_b^1$, for $b = 0,1$, $\mathsf{seed}_b^0, \mathsf{seed}_b^1 \in \{0,1\}^\lambda$ and $\mathsf{sign}_b^0, \mathsf{sign}_b^1 \in \{0,1\}$.
    Compute $\Delta\mathsf{seed}^c = \mathsf{seed}_0^c \oplus \mathsf{seed}_1^c$ and $\Delta\mathsf{sign}^c = \mathsf{sign}_0^c \oplus \mathsf{sign}_1^c$ for $c = 0,1$.
    Denote path $\leftarrow A^{(i-1)}[k]$.
    if both path$||z$ for $z = 0,1$ are in $A^{(i)}$ **then**
      $V[k] \leftarrow r || \Delta\mathsf{sign}^0 \oplus 1 || \Delta\mathsf{sign}^1 \oplus 1$, where $r \xleftarrow{\$} \{0,1\}^\lambda$.
    else
      Let z be such that path$||z \in A^{(i)}$.
      $V[k] \leftarrow \Delta\mathsf{seed}^1 || \Delta\mathsf{sign}^0 \oplus (1-z) || \Delta\mathsf{sign}^1 \oplus z$.
    end if
  end for
  return $\mathsf{OKVS}_i.\mathsf{Encode}(\{A^{(i-1)}[k], V[k]\}_{1 \leq k \leq |A^{(i-1)}|})$.
end procedure

procedure GenConvCW($A, B, \{\mathsf{seed}_b^{(n)}, \mathsf{sign}_b^{(n)}\}$)
  Sample a list V of t random $\mathbb{G}$-elements.
  for $k = 1$ to $|A|$ **do**
    $\Delta g \leftarrow G_{\mathsf{convert}}(\mathsf{seed}_0^{(n)}[k]) - G_{\mathsf{convert}}(\mathsf{seed}_1^{(n)}[k])$.
    $V[k] \leftarrow (-1)^{\mathsf{sign}_0^{(n)}[k][k]}(\Delta g - B[k])$.
  end for
  return $\mathsf{OKVS}_{\mathsf{convert}}(\{A[k], V[k]\}_{1 \leq k \leq t})$.
end procedure

procedure Correct($\bar{x}, \mathsf{sign}, CW$)
  return $C_{\mathsf{seed}} || C_{\mathsf{sign}^0} || C_{\mathsf{sign}^1} \leftarrow \mathsf{OKVS}_i.\mathsf{Decode}(CW, \bar{x})$, where C_{sign^0} and C_{sign^1} are bits.
end procedure

procedure ConvCorrect(x, sign, CW)
  return $\mathsf{OKVS}_{\mathsf{convert}}.\mathsf{Decode}(CW, x)$.
end procedure

Figure 3: The parameter l and methods' setting that turns the paradigm of DMPF in fig. 1 into the OKVS-based DMPF.