UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Erki Külaots

Two-Party Multi-Point Function Secret Sharing

Master's Thesis (30 ECTS)

Supervisor(s): Toomas Krips, PhD

Tartu 2024

Two-Party Multi-Point Function Secret Sharing

Abstract:
Multiparty computation (MPC) is an important field of cryptography that deals with protecting the privacy of data, while allowing to do computation on that data. A key part of MPC is the parties involved having correlated randomness that they can use to make the computation or the communication between themselves more efficient, while still preserving the privacy of the data. Examples of these correlations include random oblivious transfer (OT) correlations, oblivious linear-function evaluation (OLE) correlations, multiplication triples (also known as Beaver triples) and one-time truth tables. Multi-point function secret sharing (FSS) has been shown to be a great building block for (pseudo-)random correlation generation. The main question is how to construct fast and efficient multi-point FSS schemes. Here we propose a novel approach to multi-point FSS using tree structure, pseudorandom generator and systems of linear equations. Our scheme MultiFunUSLESS has similar efficiency parameters to previously proposed multi-point FSS schemes and is more efficient in the evaluation phase, this means it is the best option in some use cases. On the whole, it provides us with a new perspective, how to construct multi-point FSS schemes. In all walks of computer science efficiency is key: algorithms should be faster, take less resources and communication should be minimal and that is what we are trying to achieve with our work.

Keywords:
Function secret sharing, multi-point function, secret sharing, linear algebra, distributed point function, finite fields

CERCS: P170 Computer science, numerical analysis, systems, control

Kahe osapoolega mitmikpunktfunktsiooni saladusejaostusskeem

Lühikokkuvõte:
Turvaline ühisarvutus on tähtis krüptograafia haru, mis tegeleb privaatsete andmete töötlemisega. Üks oluline komponent turvalises ühisarvutuses on korreleeritud juhuslikkus, mis aitab osapooltel teha arvutusi efektiivsemalt või vähendada nendevahelise suhtluse mahtu, säilitades seejuures andmete privaatsust. Mõned näited sellisest korreleeritud juhuslikkusest on juhuslikud pimeedastusseosed (OT), lineaarfunktsiooni pimeväärtustamisseosed (OLE), Beaveri kolmikud ja ühekordsed tõeväärtustabelid. Mitmikpunktfunktsiooni saladusejaostust saab edukalt kasutada (pseudo)juhuslike seoste genereerimisel. Sellest tulenevalt tekib küsimus, et kuidas saame konstrueerida efektiivset mitmikpunktfunktsiooni saladusejaostusskeemi. Käesolevas magistritöös läheneme me sellele küsimusele uutmoodi, kasutades puu struktuuri, pseudojuhuarvude generaatorit ja

lineaarvõrrandisüsteeme. Meie skeem MultiFunUSLESS on efektiivsuselt võrreldav varasemate konstruktsioonidega ja on väärtustamisfaasis neist kiireim. Seega on see teatud kasutusjuhtudel parim valik. MultiFunUSLESS võimaldab meil mitmikpunktfunktsiooni saladusejaotusskeemi konstrueerimisele läheneda uutviisi. Arvutiteaduses on efektiivsus võtmesõnaks – algoritmid peaksid arvutama kiiremini, kasutama vähem ressursse ja suhtlus peaks olema minimaalne. See ongi käesoleva magistritöö eesmärk.

Võtmesõnad:
funktsiooni saladusejaostus, mitmikpunktfunktsioon, saladusejaostus, lineaaralgebra, hajuspunktfunktsioon, lõplikud korpused

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Two-party multi-point function secret sharing is a scheme, during which two parties can reveal evaluations of some multi-point function, only if they work together. Point functions are functions that evaluate to a predetermined value b at the predetermined special point a and evaluate to 0 everywhere else. Multi-point functions have some constant number of such special points.

Multi-point function secret sharing is part of the broader topic of secure multiparty computation (MPC), which is prevalent in all areas of life, where private information needs to be processed. If private information has to be processed and no single party can be trusted with the whole data, then the computation can be shared between multiple parties such that no single party learns any significant part of the private data. Examples of this include: medical information for research, handling private financial data or keys for cryptocurrencies [IEE]. Function secret sharing (FSS) was proposed to be useful in the setting of querying and updating distributed databases (private information retrieval – PIR) [BGI15]. Multi-point FSS has seen some success in the constructions of other primitives like vector oblivious linear-function evaluation (VOLE) [BCGI18] and random oblivious transfer. The common denominator is that FSS schemes can be used to make efficient pseudorandom correlation generators (PCG) and then PCG can provide correlated randomness to the parties involved. Correlated randomness is a precious resource in cryptographic applications. For example, identical random strings distributed among the parties can be used for perfectly secure encryption by using one-time pad encryption. Other examples of more useful types of random correlation in respect to MPC are random oblivious transfer (OT) correlations, oblivious linear-function correlations (OLE), multiplication triples (also known as Beaver triples [Bea91]) and one-time truth tables [BCGI18], [BCG$^+$19].

For this thesis I was given a draft for the algorithms by my supervisor. My contribution was to modify the algorithms, formalize them, prove that they are correct and secure according to the FSS definition, calculate their efficiency and compare with other multi-point FSS schemes. We call our scheme MultiFunUSLESS. Previous schemes rely on many pseudorandom generator (PRG) evaluations that are quite costly. Therefore, the idea was to try to bring the number of required PRG calls down. MultiFunUSLESS has similar key size and time complexity to other multi-point function secret sharing schemes and it shines in its simple mathematical constructions and its efficiency of evaluating the shared function at some small constant number of points.

The outline of this work is as follows. In the second section, all the preliminary knowledge that the reader needs for understanding the thesis is given. In the next section, algorithms are introduced and explained. In the fourth section the properties of MultiFunUSLESS are presented and rigorously proved. In the fifth section, prior works are listed and compared with MultiFunUSLESS. In the final section, conclusions and possible future work is given.

2 Preliminaries

This section is meant as a guide to return to, if something in later sections becomes confusing. However, it is recommended to read through sections about secret sharing, functional secret sharing, distributed point functions and multi-point functions as these topics are less well known. Most of the definitions given are classical with some minor changes to fit more tightly with this research paper.

2.1 Notation

Let us list some important notation.

- For sets X and Y, we denote $\mathcal{F}(X \to Y)$ as the family of functions $\varphi : X \to Y$.
- For $n \in \mathbb{N}$, we denote $\{0,1\}^n$ as bit strings with length n and $\{0,1\}^*$ as the set of all bit strings.
- For bit strings $x \in \{0,1\}^*$ and integers $i \in \mathbb{N}$, we denote x^i as the i-th bit of x.
- For bit strings $a, b \in \{0,1\}^*$, we denote $a||b$ as the concatenation of a and b.
- For bits $a^1, a^2 \in \{0,1\}$, we denote a^1a^2 as the concatenation of a^1 and a^2.
- For bit strings $a, x \in \{0,1\}^n$, we denote $x^1x^2 \dots x^i = a^1a^2 \dots a^i$ as the first i bits of x coinciding with the first i bits of a.
- For secret shared value τ from some abelian group $\mathbb{G}$ and a two-party additive secret sharing scheme, we denote $[\tau]_1$ and $[\tau]_2$ as the secret shares of party 1 and party 2. That is
$$[\tau]_1 + [\tau]_2 = \tau.$$
- For values a, b from some set X, we denote $[a \overset{?}{=} b]$ as the boolean function that checks the equality of two elements of X. In other words
$$[a \overset{?}{=} b] = \begin{cases} 0, & \text{if } a \neq b, \\ 1, & \text{if } a = b. \end{cases}$$

2.2 Cryptography

2.2.1 Probabilistic polynomial time (PPT) algorithm

Probabilistic polynomial time algorithms are algorithms that run for polynomial number of steps and that can use some internal randomness to make choices. Polynomial number of steps means that there exists some polynomial t_{poly} such that for all inputs $x \in \{0,1\}^*$ and all internal randomness the time it takes for the algorithm to halt is less than $t_{poly}(|x|)$ [Gol01].

2.2.2 Pseudorandom generator (PRG)

A pseudorandom generator is a polynomial time deterministic algorithm that gets a small uniform seed as its input and outputs a large pseudorandom value [KL08]. Pseudorandom here means that no bounded adversary can distinguish between this large value and a uniform large value. More formally,

Definition 1. *Let X and Y be sets such that $|X| < |Y|$. We say that a function $f : X \to Y$ is a $\varepsilon_{\mathsf{PRG}}$-**pseudorandom generator** (PRG), if for every PPT adversary $\mathcal{A}$*

$$\left|\Pr\left[\mathcal{A}(f(x)) = 1 \mid x \xleftarrow{\$} X\right] - \Pr\left[\mathcal{A}(y) = 1 \mid y \xleftarrow{\$} Y\right]\right| \leq \varepsilon_{\mathsf{PRG}}.$$

2.2.3 Secret sharing

Secret sharing is the process of sharing a piece of data between multiple parties such that a malicious subset of parties cannot access the data without cooperating with at least t parties, where t is a threshold set by the scheme [Sha79][KL23]. A piece of data held by one party is called a **secret share**. Let us denote the set of parties with $\mathcal{P} := \{P_1, \ldots, P_n\}$.

Definition 2. *A **perfect secret sharing scheme** for a set of secrets S, threshold t and set of parties $\mathcal{P}$ consists of a pair of PPT algorithms (*share, $\mathsf{reconstruct}$*), where*

- share*: On input $s \in S$ outputs shares $\sigma_1, \ldots, \sigma_n \in \{0, 1\}^*$;*
- $\mathsf{reconstruct}$*: On inputs (i, σ_i) from parties $P_i \in \hat{\mathcal{P}} \subseteq \mathcal{P}$, outputs the secret $s' \in S$ or the error symbol $\bot$.*

This scheme must satisfy the following properties:

- ***Completeness**: For every $s \in S$ and every subset of parties $\hat{\mathcal{P}} \subseteq \mathcal{P}$, that satisfy $\left|\hat{\mathcal{P}}\right| \geq t$*

  $$\Pr\left[s' = s \mid (\sigma_1, \ldots, \sigma_n) \leftarrow \mathsf{share}(s); s' \leftarrow \mathsf{reconstruct}(\{P_i() : P_i \in \hat{\mathcal{P}}\})\right] = 1,$$

  where $P_i() := (i, \sigma_i)$ denotes the output of the i-th party;

- ***Privacy**: For every $s \in S$, every subset of parties $\hat{\mathcal{P}} \subset \mathcal{P}$, that satisfy $\left|\hat{\mathcal{P}}\right| < t$, and every PPT algorithm $\mathcal{A}$*

  $$\Pr\left[s' = s \mid (\sigma_1, \ldots, \sigma_n) \leftarrow \mathsf{share}(s); s' \leftarrow \mathcal{A}(\{P_i() : P_i \in \hat{\mathcal{P}}\})\right] \leq p_{triv},$$

where $P_i() := (i, \sigma_i)$ denotes the output of the i-th party and p_{triv} denotes the trivial probability of guessing the secret or guessing the shares of other parties

$$p_{triv} := \max\{\max_{s \in S}\{\Pr[s \leftarrow S]\}, \max_{\substack{P_i \in \mathcal{P} \setminus \hat{\mathcal{P}} \\ \sigma_i \in \{0,1\}^*}} \{\Pr[\sigma_i \leftarrow \mathsf{share}(s)]\}\}.$$

Definition 3. *Let $\mathbb{G}$ be an abelian group.* ***Additive secret sharing*** *is a perfect secret sharing scheme with threshold $t = n = |\mathcal{P}|$ and $p_{triv} = \max_{s \in \mathbb{G}}\{\Pr[s \leftarrow \mathbb{G}]\}$, for which*

- share*: On input $s \in \mathbb{G}$, samples $\sigma_1, \ldots, \sigma_{n-1} \overset{\$}{\leftarrow} \mathbb{G}$. Then computes*

$$\sigma_n = s - \sum_{i=1}^{n-1} \sigma_i$$

  and outputs shares $\sigma_1, \ldots, \sigma_n$.

- reconstruct*: On input of all n shares $\sigma_1, \ldots, \sigma_n$, outputs $\sum_{i=1}^{n} \sigma_i$. On any other subset of input shares outputs $\perp$.*

2.2.4 Function secret sharing (FSS)

Function secret sharing is a type of secret sharing, where the data shared by the parties is a description of some function φ from a predefined function family $\mathcal{F}$. This description can be used to calculate the secret share of $\varphi(x)$ for any x in the domain of φ [BGI15]. More formally, let $p \in \mathbb{N}$.

Definition 4. *A* ***p-party function secret sharing (FSS) scheme*** *with respect to function class $\mathcal{F}(X \to Y)$ and secret sharing scheme SS is a pair of PPT algorithms (*Gen, Eval*), where*

- Gen*: Gets the description of $\varphi \in \mathcal{F}$ as its input and it outputs p keys $k_1, \ldots, k_p \in \{0,1\}^*$.*
- Eval*: On input (i, k_i, x), the algorithm outputs the i-th party's secret share of $\varphi(x)$ in respect to SS , where $i \in \{1, \ldots, p\}$, $k_i \in \{0,1\}^*$ and $x \in X$.*

The scheme must satisfy the following properties:

- ***Completeness****: For all $\varphi \in \mathcal{F}(X \to Y)$, $x \in X$,*

$$\Pr[\mathsf{reconstruct}(\mathsf{Eval}(1, k_1, x), \ldots, \mathsf{Eval}(n, k_p, x)) = \varphi(x) | \\ k_1, \ldots, k_p \leftarrow \mathsf{Gen}(\varphi)] = 1.$$

- ***Security**: We call an FSS scheme $(Q, \varepsilon_{\mathsf{FSS}})$-indistinguishable, if for all corrupted parties $T \subset \{1, \dots, p\}$, $|T| \leq Q < p$ and for all PPT algorithms $\mathcal{A}$:*

$$\Pr\left[b = \bar{b}\right] - \frac{1}{2} \leq \varepsilon_{\mathsf{FSS}},$$

where b and $\bar{b}$ come from the following experiment:

  - *The adversary outputs $(\varphi_1, \varphi_2, \sigma) \leftarrow \mathcal{A}()$, where $\varphi_1, \varphi_2 \in \mathcal{F}(X \to Y)$ are descriptions of two functions from function family $\mathcal{F}$ and $\sigma \in \{0,1\}^*$ is the state of $\mathcal{A}$.*
  - *The challenger selects $b \leftarrow \{0,1\}$ and calculates $(k_1, \dots, k_p) \leftarrow \mathsf{Gen}(\varphi_b)$.*
  - *The adversary gets the corrupted keys and calculates $\bar{b} \leftarrow \mathcal{A}(\{k_i\}_{i \in T}, \sigma)$.*

Note that in the following sections we denote $\mathsf{Eval}(i, k_i, x)$ as $\mathsf{Eval}^i(k_i, x)$.

2.2.5 Distributed point function (DPF)

Let X and Y be some sets e.g. $X = \{0,1\}^n$ and $Y = \{0,1\}^m$ for some $n, m \in \mathbb{N}$. The following definitions come from [BGI15].

Definition 5. *For $a \in X$ and $b \in Y$ the **point function** $P_{a,b}$ is defined by $P_{a,b}(a) = b$ and $P_{a,b}(x) = 0$ for all $x \neq a$.*

Definition 6 ([BGI15], [GI14])**.** *A **distributed point function (DPF)** is an FSS scheme with respect to the family of point functions over $X = \{0,1\}^n$ and $Y = \{0,1\}^m$.*

2.2.6 Multi-point function

Let X and Y be some sets e.g. $X = \{0,1\}^n$ and $Y = \{0,1\}^m$ for some $n, m \in \mathbb{N}$.

Definition 7 ([BCGI18])**.** *For $\vec{a} \in X^t$ and $\vec{b} \in Y^t$ the t-**multi-point function** $P_{\vec{a},\vec{b}}$ is defined by*

$$P_{\vec{a},\vec{b}}(x) = \begin{cases} b_i, & \text{if } x = a_i, \\ 0, & \text{otherwise}, \end{cases}$$

where a_i is the i-th element of $\vec{a}$ and b_i is the i-th element of $\vec{b}$.

2.2.7 Ideal Cipher Model

We define block cipher more broadly than usual.

Definition 8 ([KL08])**.** *Let $\gamma, \kappa \in \mathbb{N}$ and $F : \{0,1\}^\gamma \times \{0,1\}^\kappa \to \{0,1\}^\gamma$ be an efficient function. We call F a **keyed permutation** or a **block cipher** if for all $k \in \{0,1\}^\kappa$ the function $F(\cdot, k)$ is bijective.*

Definition 9 ([CPS08])**.** *The **Ideal Cipher Model** (ICM) is an idealized model, where a publicly accessible block cipher exists. This block cipher takes in κ bit keys and γ bit inputs and returns γ bit outputs and is chosen uniformly from all such block ciphers. This is equivalent to choosing a function from 2^κ independent random permutations.*

2.3 Statistical distance and computational distance

In this section statistical distance comes from [Gol01] and computational distance from [KL08].

Definition 10 (Probability Ensemble)**.** *Let I be a countable index set. An **ensemble indexed by** I is a sequence of random variables indexed by I. Namely any $X = \{X_i\}_{i \in I}$, where each X_i is a random variable, is an ensemble indexed by I.*

Definition 11. *Let $X := \{X_i\}_{i \in \mathbb{N}}$ and $Y := \{Y_i\}_{i \in \mathbb{N}}$ be ensembles. The **statistical distance** between X and Y is defined as*

$$SD(X, Y) = \frac{1}{2} \cdot \sum_{\alpha} \left| \Pr\left[X_n = \alpha\right] - \Pr\left[Y_n = \alpha\right] \right|.$$

Definition 12. *Let $X := \{X_i\}_{i \in \mathbb{N}}$ and $Y := \{Y_i\}_{i \in \mathbb{N}}$ be ensembles. We say X and Y are ε-**indistinguishable** if for every PPT distinguisher D:*

$$\left| \Pr\left[D(X_n) = 1\right] - \Pr\left[D(Y_n) = 1\right] \right| \leq \varepsilon,$$

where $D(X_n)$ means that x is chosen according to the distribution of X_n and then $D(X_n)$ is run.

Definition 13. *We call two cryptographic games $\mathcal{G}_0$ and $\mathcal{G}_1$ ε-**close**, if for all PPT algorithms $\mathcal{A}$*

$$\left| \Pr\left[\mathcal{G}_0(\mathcal{A}) = 1\right] - \Pr\left[\mathcal{G}_1(\mathcal{A}) = 1\right] \right| \leq \varepsilon,$$

where $\mathcal{G}_0(\mathcal{A})$ denotes that $\mathcal{A}$ is used as a subroutine in $\mathcal{G}_0$ and the probability is taken over the randomness of the games and the randomness of $\mathcal{A}$.
***Note:** Two ensembles can also be called ε-**close**, if the statistical distance between them is less or equal than ε.*

2.4 Algebra

The following definitions come from [MP13].

Definition 14. *A **ring** $(R, +, \cdot)$ is a nonempty set R with two operations "$+$"(addition) and "$\cdot$"(multiplication) such that*

1. $(R, +)$ *is an abelian group;*
2. *multiplication "$\cdot$" is associative i.e for all* $a, b, c \in R$*:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$*;*
3. *left and right distributive laws hold i.e for all* $a, b, c \in R$*:*

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{and} \quad (a + b) \cdot c = a \cdot c + b \cdot c.$$

Definition 15. *Let* $(R, +, \cdot)$ *be a ring.* R *is a* ***field*** *if it satisfies the following conditions:*

1. *multiplication "$\cdot$" is commutative i.e for all* $a, b \in R$*:* $a \cdot b = b \cdot a$*;*
2. *all non-zero elements of* R *form a group under "$\cdot$".*

Definition 16. *A* ***finite field*** *is a field with finite number of distinct elements.*

Definition 17. *If* R *is a ring and there exists such positive integer* c *such that for all* $x \in R$ *it holds* $cx = 0$*, then the smallest of such integers is called the* ***characteristic*** *of the ring* R*. Otherwise, the characteristic of the ring is* 0*.*

For example, $\mathbb{F}_{2^k}$ has characteristic 2, because addition in the field boils down to XORing of the bit representations of elements.

Now let $\mathbb{F}$ be a field and $m, n \in \mathbb{N}$. The next definitions and theorems come from [AR91].

Definition 18. *Let* $r \in \mathbb{N}$ *and* $v_1, \ldots, v_r$ *be vectors in a vector space* V *over the field* $\mathbb{F}$*. Vector* $w \in V$ *is called a* ***linear combination*** *of the vectors* $v_1, \ldots, v_r$*, if there exist* $k_1, \ldots, k_r \in \mathbb{F}$ *such that*

$$w = k_1 v_1 + \ldots + k_r v_r.$$

Definition 19. *Let* $r \in \mathbb{N}$ *and* $v_1, \ldots, v_r$ *be vectors in a vector space* V*. We call subspace* $\tilde{V}$ *of* V *the* ***space spanned by*** $v_1, \ldots, v_r$ *if every* $\tilde{v} \in \tilde{V}$ *can be expressed as a linear combination of* $v_1, \ldots, v_r$ *and every linear combination of* $v_1, \ldots, v_r$ *is in* $\tilde{V}$*.*

Definition 20. *If* A *is a* $m \times n$ *matrix over the field* $\mathbb{F}$*, then the subspace of* $\mathbb{F}^n$ *spanned by the row vectors of* A *is called the* ***row space*** *of* A*. The subspace of* $\mathbb{F}^m$ *spanned by the column vectors of* A *is called the* ***column space*** *of* A*. The solution space of homogeneous system of linear equations* $Ax = 0$ *is called the* ***nullspace*** *or the* ***kernel*** *of* A*.*

Theorem 1. *For any matrix* A *over* $\mathbb{F}$*, the dimension of the row space and the dimension of the column space are equal.*

Definition 21. *The common dimension of the row space and the column space of matrix* A *is called the* ***rank*** *of the matrix. It is denoted by* $\mathsf{rank}(A)$*.*

Definition 22. *The dimension of the nullspace of A is called* ***nullity*** *of A and is denoted by* $\mathsf{nullity}(A)$.

Theorem 2. *A system of linear equations $Ax = b$ is solvable if and only if the rank of the coefficient matrix A is the same as the rank of the augmented matrix $(A \mid b)$.*

Theorem 3 (Dimension theorem of matrices)**.** *If A is a matrix with n columns then*

$$\mathsf{rank}(A) + \mathsf{nullity}(A) = n.$$

2.5 Probability theory

In this subsection, some important results from probability theory are listed from the book [GW09].

Let A, B be events. Here $A \cap B$ corresponds to "both events occur", $A \cup B$ corresponds to "at least one event occurs" and $\neg A$ corresponds to "A does not occur". If $A \cap B = \emptyset$, then we say A and B are disjoint events.

Definition 23. *If $\Pr[B] > 0$ then the* ***conditional probability*** *of A given B is denoted as $\Pr[A \mid B]$ and defined as*

$$\Pr[A \mid B] := \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Theorem 4. *If $\{B_i \mid i \in \{1, \dots, n\}\}$, $n \in \mathbb{N}$ is a set of disjoint events such that $\Pr\left[\bigcup_{i \in \{1,\dots,n\}} B_i\right] = 1$ and for all $i, j \in \{1, \dots, n\}$ either $\Pr[B_i \cap B_j] = 0$ or $i = j$, then*

$$\Pr[A] = \sum_{i \in \{1,\dots,n\}} \Pr[A \mid B_i].$$

2.6 Big O notation

Let $f : \mathbb{N}^+ \to \mathbb{R}^+$ be a function.

Definition 24 ([MSS08])**.** *We define the following sets:*

- $O(f(n)) := \{g(n) \mid \exists c > 0, \exists n_0 \in \mathbb{N}^+, \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$,
- $\Omega(f(n)) := \{g(n) \mid \exists c > 0, \exists n_0 \in \mathbb{N}^+, \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$,
- $\Theta(f(n)) := O(f(n)) \cap \Omega(f(n))$.

These sets are useful in describing asymptotic behaviour of different functions, for example, time complexity and number of bits for communication.

2.7 Bernoulli inequality

The following result is well known and was taken from [Car00].

Theorem 5 (Bernoulli inequality)**.** *Let $a \in \mathbb{R}$, $n \in \mathbb{N}$. If $a > -1$ and $n \geq 1$, then*

$$(1-a)^n \geq 1 - na.$$

3 Algorithms

From now on let $k, n, t, v \in \mathbb{N}$, $v \geq t+1$, $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}^v$ be a $\varepsilon_{\mathsf{PRG}}$-PRG, $(a_1, \dots, a_t) \in (\{0,1\}^n)^t$ and $(b_1, \dots, b_t) \in \mathbb{F}_{2^k}^t$. We require that if $i, j \in \{1, \dots, t\}$ and $i \neq j$, then $a_i \neq a_j$ The values a_j, b_j define a multi-point function $\varphi : \{0,1\}^n \to \mathbb{F}_{2^k}$ such that for all $j \in \{1, \dots, t\}$

$$\varphi(x) = \begin{cases} b_j, & \text{if } x = a_j, \\ 0, & \text{otherwise.} \end{cases}$$

A function secret sharing scheme consists of two algorithms – the key generation algorithm (Gen) and the evaluation algorithm (Eval). The following subsections define these algorithms. The idea is to generalize DPF construction of Boyle et. al [BGI15], [BGI16] to the multi-point function case. This means the keys returned by the key generation algorithm should define complete binary trees with k levels.

To evaluate φ on x both parties essentially move bit by bit down the binary trees defined by the keys given to the parties by Gen. This means if we are at some node and the next bit is 0, then we move to the left child and if the next bit is 1, then we move right. If a path from the root node to some leaf node follows some a_j, then we call that path **an alive path**. Every other path from root to leaf is called **a dead path**. We call nodes that lie on some alive path **alive nodes** and other nodes **dead nodes**. The goal is to have a shared secret of 0 at the dead nodes and non-zero shared secret at alive nodes. For this we use some clever linear algebra. Both parties have to apply a linear function to the values at the leaf nodes to get either a shared secret of 0, if the last node was dead, or b_j, if the alive path followed was a_j. Note that every alive node has to have at least one alive child, but it can have two. Also note that there are at most t alive nodes at each level of the binary tree. Dead nodes cannot have any alive children, therefore our construction must preserve the deadness of nodes. The key part we use in accomplishing this goal is that we are working over a field of characteristic 2. This means the sum of two equal elements is always zero and if the sum of two elements is zero, then they are equal. We use a deterministic PRG to guarantee that, if both secret sharings become the same, then they stay the same.

For example, on Figure 1 alive paths are colored green, alive nodes are colored green and dead nodes are colored black and the 3-multi-point function from the function family $\mathcal{F}(\{0,1\}^4 \to \mathbb{F}_{2^3})$ is defined by points $(0010, 001)$, $(0011, 101)$ and $(1011, 010)$. On the

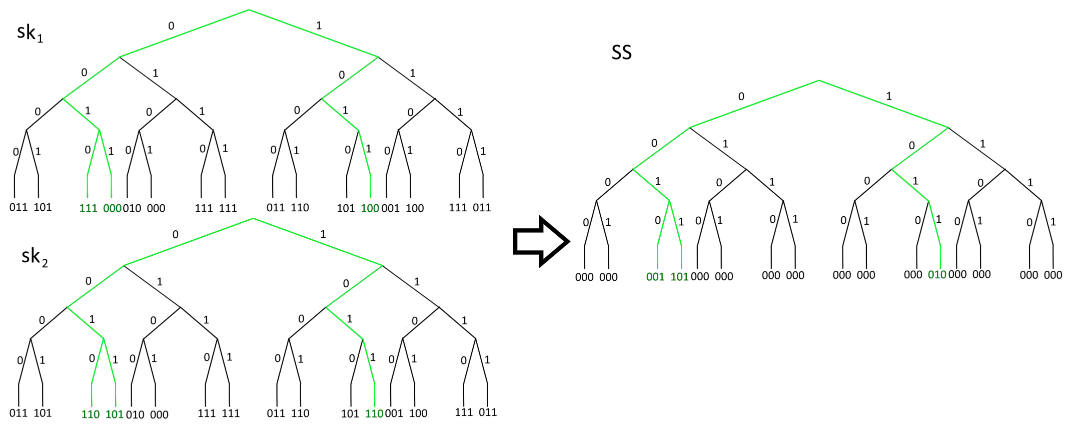

Figure 1. The main idea of MultiFunUSLESS

left we can see keys sk_1 and sk_2 and on the right we see the resulting shared secret of the sum of the keys.

3.1 Key generation algorithm

The key generation algorithm (formalised in Algorithm 1 and Algorithm 2) works as follows:

1. Sample X_ϵ from $\mathbb{F}_{2^k}^{v+1} \setminus \{0\}$ and τ_ϵ from $\mathbb{F}_{2^k}$. Sample shares of $(X_\epsilon, \tau_\epsilon)$ for party one and party two. Add ϵ to set R_0.

2. Iterate over $i \in \{1, \ldots, n\}$ and at each iteration do the following:

   (a) Randomly sample two different values $w_{i,0}$ and $w_{i,1}$ from $\mathbb{F}_{2^k}$. They have to be different, because this ensures that alive nodes with exactly one dead child can kill only one child and not both of them.

   (b) Iterate over the set R_{i-1} (set of alive nodes) and separate its nodes' children $r||b$ into alive nodes R_i and dead nodes R'_i. Here $r \in \{0,1\}^{i-1}$ and $b \in \{0,1\}$. Collect the nodes from R_{i-1} with two children into the set $\widehat{R}_{i-1}$.

   (c) For each dead node $r||b$ in R'_i, add a constraint $\left\langle X_r, \vec{d}_{i-1} \right\rangle = \tau_r \cdot w_{i,b}$ to a system of linear equations. This kills this dead node.

   (d) For each alive node r in $\widehat{R}_{i-1}$ from the previous layer $i-1$ that have two alive children $r||0$ and $r||1$, add a constraint $\left\langle X_r, \vec{d}_{i-1} \right\rangle = \tau_r \cdot w_{r,2}$ to the system of linear equations, where $w_{r,2} \in \mathbb{F}_{2^k}$ is not equal to $w_{i,0}$ nor $w_{i,1}$. This ensures with high probability, that alive nodes are not killed by accident.

(e) Solve for $\vec{d}_{i-1} \in \mathbb{F}_{2^k}^v$. Since there are at most t rows in the system of linear equations and $v > t$ variables, then we know there are either no solutions or a lot of solutions. If the system is not solvable, then return $\perp$ and abort. If it is solvable, then sample one of those solutions.

(f) For each alive node $r||b$ in R_i, set as the new secret shared values

$$([X_{r||b}]_1, [\tau_{r||b}]_1) \leftarrow f(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,b}),$$
$$([X_{r||b}]_2, [\tau_{r||b}]_2) \leftarrow f(\left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,b}),$$
$$(X_{r||b}, \tau_{r||b}) \leftarrow ([X_{r||b}]_1, [\tau_{r||b}]_1) + ([X_{r||b}]_2, [\tau_{r||b}]_2).$$

Note that we do not have to do this operation for dead nodes, because at each iteration we are only using the secret shares of the values in the alive nodes.

3. Solve linear system of equations, where for every $j \in \{1, \ldots, t\}$

$$\langle X_{a_j}, \vec{g} \rangle = b_j + \tau_{a_j},$$

and uniformly sample a solution to this. If the system is not solvable, then return $\perp$ and abort.

4. Set $([X_\epsilon]_1, [\tau_\epsilon]_1, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$ as the first key sk_1 and send it to P_1.

5. Set $([X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$ as the second key sk_2 and send it to P_2.

```
Algorithm 1: Gen – Key generator for the scheme
  Input :$\{(a_1, b_1), \dots, (a_t, b_t)\}, f$
1 $[X_\epsilon]_1 \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
2 $[X_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}^v \setminus \{[X_\epsilon]_1\}$;
3 $X_\epsilon \leftarrow [X_\epsilon]_1 + [X_\epsilon]_2$;
4 $[\tau_\epsilon]_1, [\tau_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}$;
5 $\tau_\epsilon \leftarrow [\tau_\epsilon]_1 + [\tau_\epsilon]_2$;
6 $R_0 \leftarrow \{\epsilon\}$, where $\epsilon$ is an empty string;
7 for $i = 1$ to $n$ do
8     $w_{i,0}, w_{i,1}, \vec{d}_{i-1}, R_i \leftarrow \mathsf{SubGen}()$;
9 $A \leftarrow \begin{pmatrix} X_{a_1} \\ \vdots \\ X_{a_t} \end{pmatrix}$;
10 $\vec{B} \leftarrow \begin{pmatrix} b_1 + \tau_{a_1} \\ \vdots \\ b_t + \tau_{a_t} \end{pmatrix}$;
11 Solve $A\vec{g} = \vec{B}$ and sample $\vec{g}$ from the solution space;
12 $sk_1 \leftarrow [X_\epsilon]_1, [\tau_\epsilon]_1, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g}$;
13 $sk_2 \leftarrow [X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g}$;
14 Give to $P_1$ the key $sk_1$;
15 Give to $P_2$ the key $sk_2$;
```

Algorithm 2: SubGen – Subroutine for Gen(Algorithm 1)

```
Input  : The state of Gen
Output : $w_{i,0}, w_{i,1}, \vec{d}_{i-1}, R_i$
1  $w_{i,0} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0\}$;
2  $w_{i,1} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
3  $R_i \leftarrow \emptyset$;
4  $R'_i \leftarrow \emptyset$;
5  $\widehat{R}_{i-1} \leftarrow \emptyset$;
6  for $r \in R_{i-1}$ do
7      for $b \in \{0,1\}$ do
8          if $\exists a_j$ such that $a_j$ starts with $r||b$ then
9              $R_i \leftarrow R_i \cup \{r||b\}$;
10         else
11             $R'_i \leftarrow R'_i \cup \{r||b\}$;
12     if $r||0 \in R_i$ and $r||1 \in R_i$ then
13         $\widehat{R}_{i-1} \leftarrow \widehat{R}_{i-1} \cup \{r\}$;
14 Let $A$ be a $|R_{i-1}| \times v$ matrix of zeroes;
15 Let $\vec{B}$ be a zero column vector of length $|R_{i-1}|$;
16 $j_i \leftarrow 1$;
17 for $r||b \in R'_i$ do
18     Set the $j_i$-th row of $A$ to be $X_r$;
19     Set the $j_i$-th element of $\vec{B}$ to be $\tau_r \cdot w_{i,b}$;
20     $j_i \leftarrow j_i + 1$ ;
21 for $r \in \widehat{R}_{i-1}$ do
22     $w_{r,2} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{w_{i,0}, w_{i,1}\}$;
23     Set the $j_i$-th row of $A$ to be $X_r$;
24     Set the $j_i$-th element of $\vec{B}$ to be $\tau_r \cdot w_{r,2}$;
25     $j_i \leftarrow j_i + 1$ ;
26 Solve $A\vec{d}_{i-1} = \vec{B}$ and sample $\vec{d}_{i-1}$ from the solution space;
27 for $r||b \in R_i$ do
28     $([X_{r||b}]_1, [\tau_{r||b}]_1) \leftarrow f(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,b})$;
29     $([X_{r||b}]_2, [\tau_{r||b}]_2) \leftarrow f(\left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,b})$;
30     $X_{r||b} \leftarrow [X_{r||b}]_1 + [X_{r||b}]_2$;
31     $\tau_{r||b} \leftarrow [\tau_{r||b}]_1 + [\tau_{r||b}]_2$;
32 return $w_{i,0}, w_{i,1}, \vec{d}_{i-1}, R_i$;
```

3.2 Evaluation algorithm

The evaluation algorithm for party $P \in \{1, 2\}$ on input $x = x^1||x^2||\dots||x^n \in \{0,1\}^n$ works as follows:

1. Set $(X_0, \tau_0) \leftarrow ([X_\epsilon]_P, [\tau_\epsilon]_P)$.
2. Iterate over $i \in \{1, \dots, n\}$ and do the following:
   (a) Calculate $z_i \leftarrow \left\langle X_{i-1}, \vec{d}_{i-1} \right\rangle + \tau_{i-1} \cdot w_{i,x^i}$.
   (b) Evaluate $(X_i, \tau_i) \leftarrow f(z_i)$. If $r||x^i := x^1||x^2||\dots||x^i$ is an alive node, then we have got $(X_i, \tau_i) = ([X_{r||x^i}]_P, [\tau_{r||x^i}]_P)$. If $r||x^i$ is a dead node, then r is either alive or dead. If r is alive then from $\left\langle X_r, \vec{d}_{i-1} \right\rangle = \tau_r \cdot w_{i,x^i}$, we can deduce that
   $$\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,x^i} = \left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,x^i}.$$
   This means both parties use the same argument in f and since f is deterministic, then we know that both parties have the same value as their secret shares, which means that their shared secret is 0. And since the evaluation algorithm for both parties continues exactly the same, we know that both secret shares will stay the same until the end of the protocol. If r is dead, then we know that for some ancestor of r must be dead and the same reasoning follows.
3. Calculate $z \leftarrow \langle X_n, \vec{g} \rangle + \tau_n$.

We formalise this in Algorithm 3. To make the analysis easier we can also divide the algorithm into Algorithms 4 and 5.

```
Algorithm 3: $\mathsf{Eval}^P$ – Evaluator for the scheme
  Input :$sk_P, f, x = x^1||\dots||x^n$
1 $X_0, \tau_0, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g} \leftarrow sk_P$;
2 for $i = 1$ to $n$ do
3     $z_i \leftarrow \left\langle X_{i-1}, \vec{d}_{i-1} \right\rangle + \tau_{i-1} \cdot w_{i,x^i}$;
4     $(X_i, \tau_i) \leftarrow f(z_i)$;
5 $z \leftarrow \langle X_n, \vec{g} \rangle + \tau_n$;
6 return $z$
```

```
Algorithm 4: Eval$_i^P$ – Subroutine for the evaluator
  Input   :$X_{i-1}, \tau_{i-1}, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i$
1 $z_i \leftarrow \left\langle X_{i-1}, \vec{d}_{i-1} \right\rangle + \tau_{i-1} \cdot w_{i,x^i}$;
2 $(X_i, \tau_i) \leftarrow f(z_i)$;
3 return $(X_i, \tau_i)$
```

```
Algorithm 5: Eval$^P$ – Another way to define Eval$^P$
  Input   :$sk_P, f, x = x^1 || \dots || x^n$
1 $X_0, \tau_0, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g} \leftarrow sk_P$;
2 for $i = 1$ to $n$ do
3     $(X_i, \tau_i) \leftarrow$ Eval$_i^P(X_{i-1}, \tau_{i-1}, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i)$;
4 $z \leftarrow \langle X_n, \vec{g} \rangle + \tau_n$;
5 return $z$
```

4 Properties of MultiFunUSLESS

In this thesis, we model the PRG f in the Ideal Cipher Model. Let us assume we have the ideal block cipher $F : \{0,1\}^{k(v+1)} \times \{0,1\}^k \to \{0,1\}^{k(v+1)}$. Sample $x \in \{0,1\}^{k(v+1)}$ and define an ideal PRG $\hat{f}$ as $F(x, \cdot)$. Since F is sampled randomly from all permutations and also x is uniform, then $\hat{f}$ produces uniformly random elements from $\{0,1\}^{k(v+1)}$. We can interpret these as elements of $\mathbb{F}_{2^k}^v$. In the following proofs assume that any PRG f is close enough to $\hat{f}$.

We have to prove that the following properties hold, then we can easily reason that this protocol satisfies the conditions for being an FSS scheme.

1. Algorithm 2 has a low probability of error assuming its inputs are correct.
2. If the secret shared by the parties P_1 and P_2 is $[0]$, then after one evaluation cycle by Algorithm 4 the secret is still $[0]$.
3. If $M_{i-1} \neq 0$, then $x^1 x^2 \dots x^{i-1} = a_j^1 a_j^2 \dots a_j^{i-1}$ for at least one of the a_j.
4. For all i and all j with high probability
$$(X_{a_j^1 || \dots || a_j^i}, \tau_{a_j^1 || \dots || a_j^i}) \neq 0.$$
5. Algorithm 1 has a low probability of error assuming its inputs are correct.
6. If there exists $j \in \{1, \dots, t\}$ such that $x = a_j$, then executing Algorithms 1 and 3 produces a shared secret of b_j.

7. If there does not exist $j \in \{1, \ldots, t\}$ such that $x = a_j$, then executing Algorithms 1 and 3 produces a shared secret of 0.

8. It is not possible to tell any information about a_j or b_j, for any $j \in \{1, \ldots, t\}$, from the viewpoint of one party.

4.1 Error probability of SubGen

The problem with solving systems of linear equations is that they might not be solvable. Therefore, there is a small chance that Algorithm 2 will terminate with an error.

Theorem 6. *The probability that Algorithm 2 will end with an error provided that its inputs are correct is*

$$\Pr\left[\textit{Algorithm 2 returns } \perp\right] \leq \frac{t}{(2^k)^{v-t+1}}.$$

By correct inputs we mean that the values SubGen *uses are correctly defined and for all alive nodes r of the previous level: $(X_r, \tau_r) \neq 0$.*

Proof. Let us analyse what is the probability of success and look at the system of linear equations $A\vec{d} = \vec{B}$. At first let us assume that $A \in \mathbb{F}_{2^k}^{t' \times v}$ is fixed, where $t' \leq v$. From Theorem 2 we get that $A\vec{d} = \vec{B}$ is solvable iff $\mathsf{rank}(A) = \mathsf{rank}(A \mid \vec{B})$. We know that the rank of a matrix shows the number of independent rows in the matrix. Thus, there exist $\mathsf{rank}(A)$ independent rows of A. The same rows must be independent in matrix $(A \mid \vec{B})$, because adding elements to independent vectors cannot make those vectors dependent. For $\mathsf{rank}(A) = \mathsf{rank}(A \mid \vec{B})$ to hold, the dependent rows of A must also be dependent in $(A \mid \vec{B})$. Since these rows are dependent on the independent rows, then there are $\mathsf{rank}(A)$ elements of $\vec{B}$ that we can choose freely and $t' - \mathsf{rank}(A)$ elements that are determined by them. Therefore, for a fixed A there are $(2^k)^{\mathsf{rank}(A)}$ different vectors $\vec{B}$ for which $\mathsf{rank}(A) = \mathsf{rank}(A \mid \vec{B})$ and the probability of getting such vector is thus

$$\Pr_{\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{t'}}\left[A\vec{d} = \vec{B} \text{ is solvable}\right] = \frac{(2^k)^{\mathsf{rank}(A)}}{(2^k)^{t'}}.$$

However, A is not fixed, it is also uniformly random. Therefore, if we want to calculate the probability of success we have to sum over all A. Hence

$$\Pr_{\substack{A \xleftarrow{\$} \mathbb{F}_{2^k}^{t' \times v} \\ \vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{t'}}}\left[A\vec{d} = \vec{B} \text{ is solvable}\right] =$$

$$= \Pr_{\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{t'}}\left[A\vec{d} = \vec{B} \text{ is solvable} \mid \mathsf{rank}(A) = 0\right] \cdot \Pr_{A \xleftarrow{\$} \mathbb{F}_{2^k}^{t' \times v}}\left[\mathsf{rank}(A) = 0\right] + \ldots +$$

$$+\Pr_{\vec{B}\xleftarrow{\$}\mathbb{F}_{2^k}^{t'}}\left[A\vec{d}=\vec{B}\text{ is solvable}\mid \mathsf{rank}(A)=t'\right]\cdot \Pr_{A\xleftarrow{\$}\mathbb{F}_{2^k}^{t'\times v}}\left[\mathsf{rank}(A)=t'\right]=$$

$$=\sum_{j=0}^{t'}\Pr_{\vec{B}\xleftarrow{\$}\mathbb{F}_{2^k}^{t'}}\left[A\vec{d}=\vec{B}\text{ is solvable}\mid \mathsf{rank}(A)=j\right]\cdot \Pr_{A\xleftarrow{\$}\mathbb{F}_{2^k}^{t'\times v}}\left[\mathsf{rank}(A)=j\right]=$$

$$=\sum_{j=0}^{t'}\left(\frac{(2^k)^j}{(2^k)^{t'}}\cdot\frac{\left|\{A\mid A\in\mathbb{F}_{2^k}^{t'\times v}\wedge \mathsf{rank}(A)=j\}\right|}{(2^k)^{t'\cdot v}}\right).$$

The number of A with rank j is

$$\left|\{A\mid A\in\mathbb{F}_{2^k}^{t'\times v}\wedge \mathsf{rank}(A)=j\}\right|=(2^k)^{\frac{j(j-1)}{2}}\cdot\prod_{i=0}^{j-1}\frac{((2^k)^{v-i}-1)((2^k)^{t'-i}-1)}{(2^k)^{i+1}-1}$$

according to [MP13]. Therefore success of a single call of SubGen is

$$\sum_{j=0}^{t'}\left(\frac{(2^k)^j}{(2^k)^{t'}}\cdot\frac{(2^k)^{\frac{j(j-1)}{2}}\cdot\prod_{i=0}^{j-1}\frac{((2^k)^{v-i}-1)((2^k)^{t'-i}-1)}{(2^k)^{i+1}-1}}{(2^k)^{t'\cdot v}}\right)=$$

$$=\frac{1}{(2^k)^{t'(v+1)}}\sum_{j=0}^{t'}\left((2^k)^{\frac{j(j+1)}{2}}\cdot\prod_{i=0}^{j-1}\frac{((2^k)^{v-i}-1)((2^k)^{t'-i}-1)}{(2^k)^{i+1}-1}\right),$$

where $t'=|R_{i-1}|$. Let us bound it further to show that this value is close to 1. Notice that the last element of the sum is also the biggest. Hence

$$\Pr[\text{success}]\geq\frac{(2^k)^{\frac{t'(t'+1)}{2}}}{(2^k)^{t'(v+1)}}\prod_{i=0}^{t'-1}\frac{((2^k)^{v-i}-1)((2^k)^{t'-i}-1)}{(2^k)^{i+1}-1}.$$

In the product we divide with the following elements $(2^k)^1-1, (2^k)^2-1,\ldots,(2^k)^{t'}-1$ and multiply by $(2^k)^1-1, (2^k)^2-1,\ldots,(2^k)^{t'}-1$. Thus, we get

$$\Pr[\text{success}]\geq\frac{(2^k)^{\frac{t'(t'+1)}{2}}}{(2^k)^{t'(v+1)}}\prod_{i=0}^{t'-1}\left((2^k)^{v-i}-1\right).$$

Notice that $\frac{t'(t'+1)}{2}$ is the sum of an arithmetic series from 1 to t' with step 1 (see [Gra72] for example). We have

$$\Pr[\text{success}]\geq\frac{(2^k)^{1+2+\ldots+t'}}{(2^k)^{t'v+t'}}\prod_{i=0}^{t'-1}\left((2^k)^{v-i}-1\right)=$$

$$= \frac{(2^k)^0 \cdot \ldots \cdot (2^k)^{t'}}{(2^k)^{t'v}(2^k)^{t'}} \prod_{i=0}^{t'-1} \left((2^k)^{v-i} - 1\right) =$$
$$= \prod_{j=0}^{t'-1} \frac{(2^k)^j}{(2^k)^v} \prod_{i=0}^{t'-1} \left((2^k)^{v-i} - 1\right).$$

Let us combine the two products and get

$$\Pr[\text{success}] \geq \prod_{i=0}^{t'-1} \frac{(2^k)^i}{(2^k)^v} \left((2^k)^{v-i} - 1\right) = \prod_{i=0}^{t'-1} \left(1 - \frac{1}{(2^k)^{v-i}}\right).$$

The smallest term in this product is when $i = t' - 1$, thus

$$\Pr[\text{success}] \geq \prod_{i=0}^{t'-1} \left(1 - \frac{1}{(2^k)^{v-t'+1}}\right) = \left(1 - \frac{1}{(2^k)^{v-t'+1}}\right)^{t'}.$$

Now from the Bernoulli inequality (5) we get

$$\Pr[\text{success}] \geq \left(1 - \frac{1}{(2^k)^{v-t'+1}}\right)^{t'} \geq 1 - \frac{t'}{(2^k)^{v-t'+1}}.$$

Since $t \geq t'$, then

$$\Pr[\text{success}] \geq 1 - \frac{t'}{(2^k)^{v-t'+1}} \geq 1 - \frac{t}{(2^k)^{v-t+1}}.$$

And thus the probability of error is

$$\Pr[\text{Algorithm 1 returns } \perp] \leq 1 - 1 + \frac{t}{(2^k)^{v-t+1}} = \frac{t}{(2^k)^{v-t+1}}.$$

□

4.2 Dead nodes' children stay dead

If the secret shared by the parties P_1 and P_2 is $[0]$, then after one evaluation cycle by Algorithm 4 the secret is still $[0]$. This can be stated more formally as follows.

Theorem 7. *For all $i \in \{1, \ldots, n\}$; $x^i \in \{0, 1\}$; $[X_{i-1}]_1, [X_{i-1}]_2 \in \mathbb{F}_{2^k}^v$; $[\tau_{i-1}]_1, [\tau_{i-1}]_2 \in \mathbb{F}_{2^k}$ and all key pairs generated by* Gen*: if $[X_{i-1}]_1 = [X_{i-1}]_2$ and $[\tau_{i-1}]_1 = [\tau_{i-1}]_2$ and*

$$([X_i]_1, [\tau_i]_1, [X_i]_2, [\tau_i]_2) \leftarrow \alpha_i([X_{i-1}]_1, [\tau_{i-1}]_1, [X_{i-1}]_2, [\tau_{i-1}]_2, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i),$$

then $[X_i]_1 = [X_i]_2$ and $[\tau_i]_1 = [\tau_i]_2$, where α_i is Algorithm 6 and Eval_i^P is Algorithm 4.

Algorithm 6: α_i – One cycle of multiparty computation

```
Input : $[X_{i-1}]_1, [\tau_{i-1}]_1, [X_{i-1}]_2, [\tau_{i-1}]_2, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i$
1 $([X_{i-1}]_1, [\tau_{i-1}]_1) \leftarrow \mathsf{Eval}_i^1([X_{i-1}]_1, [\tau_{i-1}]_1, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i)$;
2 $([X_{i-1}]_2, [\tau_{i-1}]_2) \leftarrow \mathsf{Eval}_i^2([X_{i-1}]_2, [\tau_{i-1}]_2, w_{i,0}, w_{i,1}, \vec{d}_{i-1}, f, x^i)$;
3 return $([X_{i-1}]_1, [\tau_{i-1}]_1, [X_{i-1}]_2, [\tau_{i-1}]_2)$
```

Proof. Let us fix $i \in \{1, \ldots, n\}$, $x^i \in \{0, 1\}$ and a key pair generated by Gen. Assume $[X_{i-1}]_1 = [X_{i-1}]_2$ and $[\tau_{i-1}]_1 = [\tau_{i-1}]_2$. Now let us calculate α_i

$$\begin{aligned}
[z_i]_1 \leftarrow & \left\langle [X_{i-1}]_1, \vec{d}_{i-1} \right\rangle + [\tau_{i-1}]_1 \cdot w_{i,x^i}, \\
([X_i]_1, [\tau_i]_1) \leftarrow & f([z_i]_1), \\
[z_i]_2 \leftarrow & \left\langle [X_{i-1}]_2, \vec{d}_{i-1} \right\rangle + [\tau_{i-1}]_2 \cdot w_{i,x^i} = \\
= & \left\langle [X_{i-1}]_1, \vec{d}_{i-1} \right\rangle + [\tau_{i-1}]_1 \cdot w_{i,x^i} = [z_i]_1, \\
([X_i]_2, [\tau_i]_2) \leftarrow & f([z_i]_2) = f([z_i]_1) = ([X_i]_1, [\tau_i]_1).
\end{aligned}$$

Therefore, we get

$$[X_i]_1 = [X_i]_2, \quad \text{and} \quad [\tau_i]_1 = [\tau_i]_2,$$

which is exactly what we wanted to show. □

4.3 Node is alive if it is supposed to be alive

This can be stated more formally as follows.

Theorem 8. *For all $i \in \{0, \ldots, n\}$ if $M_i \neq 0$, then there exists $j \in \{1, \ldots, t\}$ such that*

$$x^1 x^2 \ldots x^i = a_j^1 a_j^2 \ldots a_j^i.$$

Notice that due to the construction of Algorithms 1 and 3, we can be sure, that the shared value in the evaluation phase (X_{i-1}, τ_{i-1}) is the same as $(X_{x^1 x^2 \ldots x^{i-1}}, \tau_{x^1 x^2 \ldots x^{i-1}})$ in the key generation phase. This is the reason why we can denote M_{i-1} as $(X_{x^1 x^2 \ldots x^{i-1}}, \tau_{x^1 x^2 \ldots x^{i-1}})$.

Proof. Let us prove this by induction.

Base step: Let $i = 0$. The statement holds trivially, because there is no 0-th bit of x, therefore, it coincides with the beginning of every a_j.

Induction step: Let us assume for $i - 1$ the statement holds and denote

$$r := x^1 x^2 \ldots x^{i-1},$$

i.e.

$$(X_r, \tau_r) \neq 0 \quad \Rightarrow \quad \exists j \in \{1, \dots, t\} : r = a_j^1 a_j^2 \dots a_j^{i-1}.$$

Let us show the statement holds for i as well and assume towards contradiction that

$$(X_{r||x^i}, \tau_{r||x^i}) \neq 0 \quad \wedge \quad \nexists j \in \{1, \dots, t\} : r||x^i = a_j^1 a_j^2 \dots a_j^i.$$

Since $(X_{r||x^i}, \tau_{r||x^i}) \neq 0$, then property 2 implies that $(X_r, \tau_r) \neq 0$. Now we can use the induction assumption and get that $\exists j \in \{1, \dots, t\} : r = a_j^1 a_j^2 \dots a_j^{i-1}$. This means that during the execution of SubGen $r \in R_{i-1}$ and since $\nexists j \in \{1, \dots, t\} : r||x^i = a_j^1 a_j^2 \dots a_j^i$, then we know that $r||x^i \in R_i'$. In SubGen lines 17 to 20 the following equation is enforced:

$$\left\langle X_r, \vec{d}_{i-1} \right\rangle = \tau_r \cdot w_{i,x^i},$$

which, because of linearity of the scalar product and the fact that

$$[X_r]_1 + [X_r]_2 = X_r \quad \text{and} \quad [\tau_r]_1 + [\tau_r]_2 = \tau_r,$$

is the same as

$$\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + \left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle = [\tau_r]_1 \cdot w_{i,x^i} + [\tau_r]_2 \cdot w_{i,x^i},$$

that is

$$\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle - [\tau_r]_1 \cdot w_{i,x^i} = - \left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,x^i}.$$

Since we are working in $\mathbb{F}_{2^k}$, then we get

$$\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,x^i} = \left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,x^i}.$$

Now we can calculate $(X_{r||x^i}, \tau_{r||x^i})$:

$$\begin{aligned}(X_{r||x^i}, \tau_{r||x^i}) &= ([X_{r||x^i}]_1, [\tau_{r||x^i}]_1) + ([X_{r||x^i}]_2, [\tau_{r||x^i}]_2) = \\ &= f\left(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,x^i}\right) + f\left(\left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,x^i}\right) = \\ &= f\left(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,x^i}\right) + f\left(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,x^i}\right) = \\ &= (0, \dots, 0, 0).\end{aligned}$$

This is a contradiction, because we assumed $(X_{r||x^i}, \tau_{r||x^i}) \neq 0$. Therefore, there exists $j \in \{1, \dots, t\} : r||x^i = a_j^1 a_j^2 \dots a_j^i$. By the principles of mathematical induction, the statement holds for any $i \in \{0, \dots, n\}$. $\square$

4.4 Alive nodes are probably not dead

Theorem 9. *For all $i \in \{0, \dots, n\}$, for all $j \in \{1, \dots, t\}$ and for all PRG $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}^{v+1}$ for which*

$$\Pr\left[f(x) = f(x') \wedge x \neq x' : x, x' \xleftarrow{\$} \mathbb{F}_{2^k}\right] \leq \varepsilon_{col},$$

it holds

$$\Pr\left[(X_{a_j^1||\dots||a_j^i}, \tau_{a_j^1||\dots||a_j^i}) \neq 0\right] \geq \left(p_{solvable} \cdot (1 - \varepsilon_{col})^t\right)^i,$$

where the probability is taken over the randomness of Gen *and*

$$p_{solvable} := 1 - \frac{t}{(2^k)^{v-t+1}}.$$

Proof. Let us prove this claim by induction over $i \in \{0, \dots, n\}$.

Base step: Let $i = 0$. In any case $(X_\epsilon, \tau_\epsilon) \neq 0$ and thus the probability is 1, because the construction of Gen ensures that X_ϵ is not 0. Therefore the claim holds trivially.

Induction step: Let us assume that the claim holds for $i - 1$ and show it also holds for i. Denote

$$p_i := \Pr\left[(X_{a_j^1||\dots||a_j^i}, \tau_{a_j^1||\dots||a_j^i}) \neq 0\right],$$

and $r_j := a_j^1||\dots||a_j^{i-1}$.

The proof follows the scheme seen on Figure 2. Nodes of the scheme represent different events and edges between them represent the probability of this event happening on the condition that previous events happened. If there is only one outgoing edge, then the next event happened with probability 1 given the previous events. If there are two outgoing edges, then the sum of the probabilities equals 1, or in other words, the child events of the parent are complements of each other.

Let us start to calculate the probability of the success path from Figure 2. The first question is, if $(X_{r_j}, \tau_{r_j}) \neq 0$ for all $j \in \{1, \dots, t\}$. This happens by the induction assumption with probability

$$p_{i-1} \geq \left(p_{solvable} \cdot (1 - \varepsilon_{col})^t\right)^{i-1}.$$

On the other hand, if there exists $j \in \{1, \dots, t\}$ such that $(X_{r_j}, \tau_{r_j}) = 0$, then we know from property 2, that $(X_{r_j||a_j^i}, \tau_{r_j||a_j^i}) = 0$. Therefore, in this case we fail.

Now let us assume for all $j \in \{1, \dots, t\} : (X_{r_j}, \tau_{r_j}) \neq 0$. The next question is, whether $A\vec{d}_{i-1} = \vec{B}$ is solvable during the i-th step. If it is not, then Gen aborts and we end in failure. We know from Subsection 4.1 that it is solvable with probability at least

$$p_{solvable} := 1 - \frac{t}{(2^k)^{v-t+1}}.$$

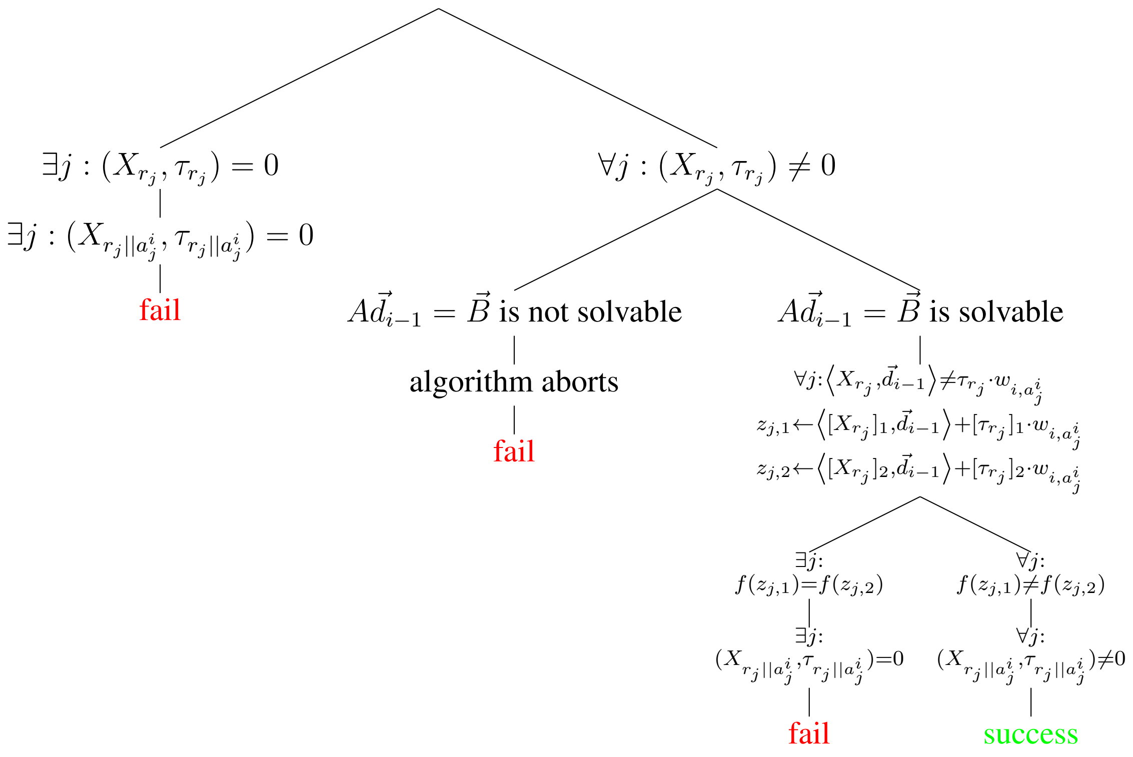
$\exists j : (X_{r_j}, \tau_{r_j}) = 0$
$\exists j : (X_{r_j||a^i_j}, \tau_{r_j||a^i_j}) = 0$
fail
$\forall j : (X_{r_j}, \tau_{r_j}) \neq 0$
$A\vec{d}_{i-1} = \vec{B}$ is not solvable
algorithm aborts
fail
$A\vec{d}_{i-1} = \vec{B}$ is solvable
$\forall j{:}\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle \neq \tau_{r_j} \cdot w_{i,a^i_j}$
$z_{j,1} \leftarrow \left\langle [X_{r_j}]_1, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_1 \cdot w_{i,a^i_j}$
$z_{j,2} \leftarrow \left\langle [X_{r_j}]_2, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_2 \cdot w_{i,a^i_j}$
$\exists j{:}$ $f(z_{j,1}) = f(z_{j,2})$
$\exists j{:}$ $(X_{r_j||a^i_j}, \tau_{r_j||a^i_j}) = 0$
fail
$\forall j{:}$ $f(z_{j,1}) \neq f(z_{j,2})$
$\forall j{:}$ $(X_{r_j||a^i_j}, \tau_{r_j||a^i_j}) \neq 0$
success

Figure 2. Proof scheme for property 4.

If $A\vec{d}_{i-1} = \vec{B}$ is solvable, then a random $\vec{d}_{i-1}$ is picked. Let us fix this value.

Let us fix $j \in \{1, \dots, t\}$. If r has one alive and one dead child, then we know that $r||a_j^i$ is alive and $r||(1 - a_j^i)$ is dead. Therefore, the system of linear equations enforces

$$\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle = \tau_{r_j} \cdot w_{i,(1-a_j^i)}.$$

Since $w_{i,(1-a_j^i)} \neq w_{i,a_j^i}$, then $\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle \neq \tau_{r_j} \cdot w_{i,a_j^i}$.

If r has two alive children, then we know that the system of linear equations enforces

$$\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle = \tau_{r_j} \cdot w_{r_j,2}.$$

Since $w_{r_j,2} \neq w_{i,a_j^i}$, then $\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle \neq \tau_{r_j} \cdot w_{i,a_j^i}$.

Therefore, we know that for all $j \in \{1, \dots, t\}$ it holds $\left\langle X_{r_j}, \vec{d}_{i-1} \right\rangle \neq \tau_{r_j} \cdot w_{i,a_j^i}$. This also means that

$$\left\langle [X_{r_j}]_1, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_1 \cdot w_{i,a_j^i} \neq \left\langle [X_{r_j}]_2, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_2 \cdot w_{i,a_j^i}.$$

Let us denote

$$z_{j,1} := \left\langle [X_{r_j}]_1, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_1 \cdot w_{i,a_j^i},$$
$$z_{j,2} := \left\langle [X_{r_j}]_2, \vec{d}_{i-1} \right\rangle + [\tau_{r_j}]_2 \cdot w_{i,a_j^i}.$$

During the execution of Gen, $(X_{r_j||a_j^i}, \tau_{r_j||a_j^i})$ is calculated as $f(z_{j,1}) + f(z_{j,2})$. Now there are two possibilities: on the one hand, if for all $j \in \{1, \dots, t\} : f(z_{j,1}) \neq f(z_{j,2})$, which means $(X_{r_j||a_j^i}, \tau_{r_j||a_j^i}) \neq 0$, then we succeed. Otherwise, if there exists $j \in \{1, \dots, t\}$ such that $f(z_{j,1}) = f(z_{j,2})$, then for that j the equality $(X_{r_j||a_j^i}, \tau_{r_j||a_j^i}) = 0$ holds. This is a failure.

What is the probability that for all $j \in \{1, \dots, t\} : f(z_{j,1}) \neq f(z_{j,2})$?
It is quite natural to think that random collisions in a PRG are very rare. Therefore, we require from our PRG $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}^{v+1}$ that

$$\Pr\left[f(x) = f(x') \wedge x \neq x' : x, x' \xleftarrow{\$} \mathbb{F}_{2^k} \right] \leq \varepsilon_{col}.$$

Let us fix $j \in \{1, \dots, t\}$. We know that $[\tau_{r_j}]_1$ and $[\tau_{r_j}]_2$ are both uniform over $\mathbb{F}_{2^k}$. Therefore, $[\tau_{r_j}]_1 \cdot w_{i,a_j^i}$ and $[\tau_{r_j}]_2 \cdot w_{i,a_j^i}$ are uniform over $\mathbb{F}_{2^k}$, because w_{i,a_j^i} is fixed and non zero. And thus $z_{j,1}$ and $z_{j,2}$ are also uniform over $\mathbb{F}_{2^k}$. From before we get $z_{j,1} \neq z_{j,2}$. This is exactly the situation that we assumed was frequent. Thus, for one j the probability of $f(z_{j,1}) \neq f(z_{j,2})$ is $1 - \varepsilon_{col}$. Assuming all t instances of $z_{j,1}, z_{j,2}$ are independent, the probability that we are looking for is equal or larger than $(1 - \varepsilon_{col})^t$.

Now let us follow the success path:

$$\begin{aligned}
p_i &\geq p_{i-1} \cdot p_{solvable} \cdot (1-\varepsilon_{col})^t \geq \\
&\geq \left(p_{solvable} \cdot (1-\varepsilon_{col})^t\right)^{i-1} \cdot p_{solvable} \cdot (1-\varepsilon_{col})^t = \left(p_{solvable} \cdot (1-\varepsilon_{col})^t\right)^i \geq \\
&\geq \left((1-\varepsilon_{col})^t \cdot \left(1 - \frac{t}{(2^k)^{v-t+1}}\right)\right)^i .
\end{aligned}$$

Therefore, by the principles of mathematical induction for all $i \in \{0, \ldots, n\}$ the statement holds. □

4.5 Error probability of Gen

In the previous subsection we saw, what is the success probability of SubGen in the sense that it does not kill alive nodes and it does not abort. These are the success conditions of Gen as well. However, in the end Gen calculates $\vec{g}$ as well, which is also a possible point of failure. This subsection captures the entire failure probability of MultiFunUSLESS. At first, let us show that, if during the process an alive node has been killed, then Gen aborts (unless that alive path is supposed to evaluate to 0).

Lemma 1. *For all $j \in \{1, \ldots, t\}$, if*

$$(X_{a_j}, \tau_{a_j}) = 0 \quad \textit{and} \quad b_j \neq 0,$$

then Gen *aborts.*

Proof. Let us fix $j \in \{1, \ldots, t\}$ and assume

$$(X_{a_j}, \tau_{a_j}) = 0 \quad \text{and} \quad b_j \neq 0.$$

During the execution of Gen at lines 9 to 11, we try to find $\vec{g} \in \mathbb{F}_{2^k}^v$ such that

$$\langle X_{a_j}, \vec{g} \rangle = b_j + \tau_{a_j}.$$

That is

$$\begin{aligned}
\langle 0, \vec{g} \rangle &= b_j + 0, \\
0 &= b_j.
\end{aligned}$$

This is not possible, because $b_j \neq 0$. Therefore, Gen aborts. □

From the previous proof we also get that, if $b_j = 0$, then Gen does not abort, due to this alive node having been killed. This is not a problem, because the end result of the protocol is still correct. The evaluation of desired t-multi-point function on $x = a_j$ is $b_j = 0$.

Theorem 10. *The probability that Algorithm 1 will end with an error provided that its inputs are correct is*

$$\Pr\left[\textit{Algorithm 1 returns } \perp\right] \leq 1 - p_{\textit{solvable}}^{n+1} \cdot (1-\varepsilon_{\textit{col}})^{tn},$$

where ε_{col} is the probability that two random different inputs $x_1 \neq x_2$ to the PRG f collide ($f(x_1) = f(x_2)$) and

$$p_{\textit{solvable}} := 1 - \frac{t}{(2^k)^{v-t+1}}.$$

Proof. From Lemma 1 we can conclude that Gen succeeds if Gen does not abort. By succeeding we mean that Gen does not make any keys that do not evaluate to the original multi-point function, It can happen during the executions of SubGen or during the solving of the linear equation $A\vec{g} = \vec{B}$. There are two possibilities:

1. For all $j \in \{1, \ldots, t\}$ we have

$$(X_{a_j}, \tau_{a_j}) \neq 0.$$

   This holds with probability

$$\Pr\left[(X_{a_j}, \tau_{a_j}) \neq 0\right] \geq \left(p_{\textit{solvable}} \cdot (1-\varepsilon_{\textit{col}})^t\right)^n.$$

   It is reasonable to assume that in this case (X_{a_j}, τ_{a_j}) are uniformly random for all $j \in \{1, \ldots, t\}$. The question is what is the probability of solving $A\vec{g} = \vec{B}$, if A and $\vec{B}$ are uniformly random. Thus, we have the same case as in Subsection 4.1 and the probability not aborting is greater than $p_{\textit{solvable}}$.

2. There exists $j \in \{1, \ldots, t\}$ such that

$$(X_{a_j}, \tau_{a_j}) = 0.$$

   We saw in Lemma 1 that in this case if $b_j \neq 0$, then the algorithm aborts. We do not know what is the probability of $b_j \neq 0$. Thus, it is easier to bound the success with 0.

From previous two branches we get that the probability of success is

$$\Pr\left[\mathsf{Gen\ succeeds}\right] \geq p_{\textit{solvable}}\left(p_{\textit{solvable}} \cdot (1-\varepsilon_{\textit{col}})^t\right)^n + 0 = p_{\textit{solvable}}^{n+1} \cdot (1-\varepsilon_{\textit{col}})^{tn}.$$

And the probability of aborting is

$$\Pr\left[\mathsf{Gen\ outputs} \perp\right] \leq 1 - p_{\textit{solvable}}^{n+1} \cdot (1-\varepsilon_{\textit{col}})^{tn}.$$

□

4.6 The evaluation of an alive path produces correct output

Theorem 11. *For all $x \in \{0,1\}^n$, if there exists $j \in \{1,\dots,t\}$ such that $x = a_j$, then $[b_j]_1 + [b_j]_2 = b_j$, where $[b_j]_1$ is the output of* Eval^1 *and $[b_j]_2$ is the output of* Eval^2*, provided that* Gen *succeeds.*

Proof. Let us fix $x \in \{0,1\}^n$ and assume there exists $j \in \{1,\dots,t\}$ such that $x = a_j$ and that Gen, Eval^1 and Eval^2 succeed. Let $[b_j]_1$ be the output of Eval^1 and $[b_j]_2$ be the output of Eval^2.

$$\begin{aligned}
[b_j]_1 &= \langle [X_x]_1, \vec{g} \rangle + [\tau_x]_1, \\
[b_j]_2 &= \langle [X_x]_2, \vec{g} \rangle + [\tau_x]_2.
\end{aligned}$$

Since the scalar product is linear,

$$\begin{aligned}
b_j &= \langle X_x, \vec{g} \rangle + \tau_x = \langle [X_x]_1 + [X_x]_2, \vec{g} \rangle + [\tau_x]_1 + [\tau_x]_2 = \\
&= \langle [X_x]_1, \vec{g} \rangle + [\tau_x]_1 + \langle [X_x]_2, \vec{g} \rangle + [\tau_x]_2 = [b_j]_1 + [b_j]_2.
\end{aligned}$$

Therefore, if the protocol succeeds, then P_1 and P_2 have a shared secret of b_j. □

4.7 The evaluation of a dead path produces shared secret of 0

Theorem 12. *For all $x \in \{0,1\}^n$, if there does not exists $j \in \{1,\dots,t\}$ such that $x = a_j$, then $[z]_1 + [z]_2 = 0$, where $[z]_1$ is the output of* Eval^1 *and $[z]_2$ is the output of* Eval^2*, provided that* Gen *succeeds.*

Proof. Let us fix $x \in \{0,1\}^n$ and assume there does not exists $j \in \{1,\dots,t\}$ such that $x = a_j$ and that Gen, Eval^1 and Eval^2 succeed. Let $[z_j]_1$ be the output of Eval^1 and $[z_j]_2$ be the output of Eval^2. We know from Gen and property 2, that at some point the shared secret becomes 0 and it stays 0, thus we know that $[X_x]_1 = [X_x]_2$ and $[\tau_x]_1 = [\tau_x]_2$. Let us calculate $[z_j]_1 + [z_j]_2$:

$$\begin{aligned}
[z_j]_1 + [z_j]_2 &= \langle [X_x]_1, \vec{g} \rangle + [\tau_x]_1 + \langle [X_x]_2, \vec{g} \rangle + [\tau_x]_2 = \\
&= \langle [X_x]_1, \vec{g} \rangle + [\tau_x]_1 + \langle [X_x]_1, \vec{g} \rangle + [\tau_x]_1 = 0.
\end{aligned}$$

This is exactly what we wanted to show. □

4.8 One party cannot evaluate the secret function alone

For readability, the games are moved to Subsection 4.11.

It is not possible to tell any information about a_j or b_j, for any $j \in \{1,\dots,t\}$, from the viewpoint of one party. We can define a security definition that should capture property 8.

Definition 25 (FSS real or random indistinguishability)**.** *We call a FSS scheme $\varepsilon_{\mathsf{FSS}}$-ROR-indistinguishable if for every PPT adversary $\mathcal{A}$ it holds*

$$|\Pr[\textit{Game } 1(\mathcal{A}) = 1] - \Pr[\textit{Game } 3(\mathcal{A}) = 1]| \leq \varepsilon_{\mathsf{FSS}},$$

and

$$|\Pr[\textit{Game } 2(\mathcal{A}) = 1] - \Pr[\textit{Game } 3(\mathcal{A}) = 1]| \leq \varepsilon_{\mathsf{FSS}},$$

where $\overset{P_1}{\leftarrow}$ denotes the output that is given to P_1 and $\overset{P_2}{\leftarrow}$ denotes the output that is given to P_2.

Theorem 13. *For all $\varepsilon_{\mathsf{PRG}}$-PRG $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}^{v+1}$:*
MultiFunUSLESS is $(t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat})$-ROR-indistinguishable, where

$$\varepsilon_{mat} = \frac{t}{(2^k)^{v-t+1}}.$$

Proof. In this proof, we color changes in code magenta.
Without loss of generality, let us assume that adversary $\mathcal{A}$ is playing as P_2. Also, let us assume we have $\varepsilon_{\mathsf{PRG}}$-PRG $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}^{v+1}$. This means that for every PPT adversary $\mathcal{B}$

$$|\Pr[\text{Game } 4(\mathcal{B}) = 1] - \Pr[\text{Game } 5(\mathcal{B}) = 1]| \leq \varepsilon_{\mathsf{PRG}}.$$

Now let us look at Game 2 and the first occurrence of the line (1)

$$([X_{r||b}]_1, [\tau_{r||b}]_1) \leftarrow f(\left\langle [X_r]_1, \vec{d}_{i-1} \right\rangle + [\tau_r]_1 \cdot w_{i,b}). \tag{1}$$

Since $\epsilon \in R_0$, then at least one of the corresponding node's children must be in R_1. Let us denote this child by b. Now let us look at $\left\langle [X_\epsilon]_1, \vec{d}_0 \right\rangle + [\tau_\epsilon]_1 \cdot w_{1,b}$. Since $w_{1,b} \neq 0$ and $[\tau_\epsilon]_1$ is uniform over $\mathbb{F}_{2^k}$, then $[\tau_\epsilon]_1 \cdot w_{1,b}$ is also uniform over $\mathbb{F}_{2^k}$. From this we can also conclude that $\left\langle [X_\epsilon]_1, \vec{d}_0 \right\rangle + [\tau_\epsilon]_1 \cdot w_{1,b}$ is also uniform over $\mathbb{F}_{2^k}$. Notice that $\mathcal{A}$ does not know this value, because they do not know $[\tau_\epsilon]_1$. Even if they know all the other parts of this expression $[\tau_\epsilon]_1$ hides it.

Therefore, we have reached Game 4. We can replace the first occurrence of line (1) with

$$([X_{r||b}]_1, [\tau_{r||b}]_1) \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^{v+1},$$

and get a new game called Game 2-1, which differs by only this line and is $\varepsilon_{\mathsf{PRG}}$-close to Game 2. Repeating this procedure at every occurrence of line (1) or in other words at every alive node, we can replace all such lines and reach Game 2-t', which is $t \cdot n \cdot \varepsilon_{\mathsf{PRG}}$-close to Game 2. Let us write out Game 2-t' and rename it to Game 6. This can be further simplified, because $X_{r||b}$ and $\tau_{r||b}$ on lines 30 and 31 of SubGen are now also uniformly random. We get the new game Game 7. We can do the sampling right before we need sampled values and get Game 8, which turns into Game 9, because the aim of the sampling is to sample random matrices and vectors. **Note** that the symbol "…" denotes the lines that have not been changed and are the same as in Game 2.

Since $\vec{d}_{i-1}$ is not used for calculating $X_{r||b}, \tau_{r||b}$, then we can try to replace it with a random vector of length v. Let us fix $\vec{d} \in \mathbb{F}_{2^k}^v$. Let us show that the statistical distance between Game 10 and Game 12 is small. For this let us consider Game 11. This game is like Game 10, but it returns $\perp$, if $A\vec{d} = \vec{B}$ is not solvable. In fact because of Theorem 2, Game 10 returns $\perp$ if and only if Game 11 returns $\perp$. Thus,

$$SD(\text{Game 10}, \text{Game 12}) = SD(\text{Game 11}, \text{Game 12}).$$

Let us calculate $SD(\text{Game 11}, \text{Game 12})$:

$$\begin{aligned}
&SD(\text{Game 11}, \text{Game 12}) = \\
&\quad = \frac{1}{2} \sum_{\alpha \in \mathbb{F}_{2^k}^v \cup \{\perp\}} \left| \Pr\left[\text{Game 11} = \alpha\right] - \Pr\left[\text{Game 12} = \alpha\right] \right| = \\
&\quad = \frac{1}{2} \sum_{\alpha \in \mathbb{F}_{2^k}^v} \left| \sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|}} \Pr[A] \Pr\left[\vec{B} \mid A\right] \cdot \right. \\
&\qquad\qquad \left. \cdot \Pr_{\substack{\vec{d} \xleftarrow{\$} \text{solutions to} \\ A\vec{d} = \vec{B}}} \left[\vec{d} = \alpha \mid \vec{B} \cap A\right] - \frac{1}{(2^k)^v} \right| + \\
&\quad + \frac{1}{2} \sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|}} \Pr[A] \Pr\left[\vec{B} \mid A\right] \Pr_{\substack{\vec{d} \xleftarrow{\$} \text{solutions to} \\ A\vec{d} = \vec{B}}} \left[\vec{d} = \perp \mid \vec{B} \cap A\right].
\end{aligned}$$

Since $\vec{B}$ is independent of A, then

$$\Pr[A] = \frac{1}{(2^k)^{|R_{i-1}| \cdot v}}, \quad \Pr\left[\vec{B}\right] = \frac{1}{(2^k)^{|R_{i-1}|}}.$$

Now we can see that $\vec{d} = \perp$ iff $\mathsf{rank}(A) < \mathsf{rank}(A|\vec{B})$. The analysis for this probability can be seen in Section 4.1. Therefore, the second summand is

$$\begin{aligned}
&\frac{1}{2} \cdot \sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|}} \Pr[A] \Pr\left[\vec{B}\right] \Pr_{\substack{\vec{d} \xleftarrow{\$} \text{solutions to} \\ A\vec{d} = \vec{B}}} \left[\vec{d} = \perp \mid \vec{B} \cap A\right] = \\
&= \frac{1}{2} \cdot \sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\substack{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|} \\ \mathsf{rank}(A) < \mathsf{rank}(A|\vec{B})}} \Pr[A] \Pr\left[\vec{B}\right] = \frac{1}{2} \cdot \frac{\#\text{ of } A \text{ and } \vec{B} \text{ such that } \mathsf{rank}(A) < \mathsf{rank}(A|\vec{B})}{\#\text{ of } A \text{ and } \vec{B}} =
\end{aligned}$$

$$= \frac{1}{2} \Pr\left[A\vec{d} = \vec{B} \text{ is not solvable}\right].$$

Let us fix α and look at the first summand. More precisely let us calculate

$$\sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|}} \Pr[A] \Pr\left[\vec{B}\right] \Pr_{\substack{\vec{d} \overset{\$}{\leftarrow} \text{solutions to} \\ A\vec{d} = \vec{B}}}\left[\vec{d} = \alpha \mid \vec{B} \cap A\right].$$

If $A\alpha \neq \vec{B}$, then we know that $\Pr_{\substack{\vec{d} \overset{\$}{\leftarrow} \text{solutions to} \\ A\vec{d} = \vec{B}}}\left[\vec{d} = \alpha \mid \vec{B} \cap A\right] = 0$. If $A\alpha = \vec{B}$, then $A\vec{d} = \vec{B}$ is solvable and there are $(2^k)^{v - \mathsf{rank}(A)}$ solutions to it. The probability of uniformly sampling $\vec{d}$ such that $\vec{d} = \alpha$ is thus

$$\Pr_{\substack{\vec{d} \overset{\$}{\leftarrow} \text{solutions to} \\ A\vec{d} = \vec{B}}}\left[\vec{d} = \alpha \mid \vec{B} \cap A\right] = \frac{1}{(2^k)^{v - \mathsf{rank}(A)}}.$$

We get

$$\sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|}} \Pr[A] \Pr\left[\vec{B}\right] \Pr_{\substack{\vec{d} \overset{\$}{\leftarrow} \text{solutions to} \\ A\vec{d} = \vec{B}}}\left[\vec{d} = \alpha \mid \vec{B} \cap A\right] =$$

$$= \sum_{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v}} \sum_{\substack{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|} \\ A\alpha = \vec{B}}} \frac{1}{(2^k)^{|R_{i-1}|(v+1)}} \cdot \frac{1}{(2^k)^{v - \mathsf{rank}(A)}} =$$

$$= \sum_{j=0}^{|R_{i-1}|} \left(\sum_{\substack{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v} \\ \mathsf{rank}(A) = j}} \sum_{\substack{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|} \\ A\alpha = \vec{B}}} \frac{1}{(2^k)^{|R_{i-1}|(v+1)}} \cdot \frac{1}{(2^k)^{v-j}} \right).$$

If A and α are fixed, then there is exactly one $\vec{B}$ for which $A\alpha = \vec{B}$. Therefore,

$$\sum_{j=0}^{|R_{i-1}|} \left(\sum_{\substack{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v} \\ \mathsf{rank}(A) = j}} \sum_{\substack{\vec{B} \in \mathbb{F}_{2^k}^{|R_{i-1}|} \\ A\alpha = \vec{B}}} \frac{1}{(2^k)^{|R_{i-1}|(v+1)}} \cdot \frac{1}{(2^k)^{v-j}} \right) =$$

$$= \sum_{j=0}^{|R_{i-1}|} \left(\sum_{\substack{A \in \mathbb{F}_{2^k}^{|R_{i-1}| \times v} \\ \mathsf{rank}(A) = j}} \frac{1}{(2^k)^{|R_{i-1}|(v+1)}} \cdot \frac{1}{(2^k)^{v-j}} \right) =$$

$$= \sum_{j=0}^{|R_{i-1}|} \left(\frac{\left|\{A \mid A \in \mathbb{F}_{2^k}^{|R_{i-1}|\times v} \wedge \mathsf{rank}(A) = j\}\right|}{(2^k)^{|R_{i-1}|(v+1)} \cdot (2^k)^{v-j}} \right) =$$
$$= \frac{1}{(2^k)^v} \sum_{j=0}^{|R_{i-1}|} \left((2^k)^j \cdot \frac{\left|\{A \mid A \in \mathbb{F}_{2^k}^{|R_{i-1}|\times v} \wedge \mathsf{rank}(A) = j\}\right|}{(2^k)^{|R_{i-1}|(v+1)}} \right) =$$
$$= \frac{1}{(2^k)^v} \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right],$$

where the last equality comes from Subsection 4.1. Replacing this result into the first summand we get

$$\frac{1}{2} \sum_{\alpha \in \mathbb{F}_{2^k}^v} \left| \frac{1}{(2^k)^v} \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right] - \frac{1}{(2^k)^v} \right| =$$
$$= \frac{1}{2} \cdot \frac{1}{(2^k)^v} \sum_{\alpha \in \mathbb{F}_{2^k}^v} \left| \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right] - 1 \right| =$$
$$= \frac{1}{2} \cdot \frac{1}{(2^k)^v} \cdot (2^k)^v \cdot \left(1 - \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right]\right) =$$
$$= \frac{1}{2} \cdot \left(1 - \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right]\right).$$

We can now calculate the statistical distance

$$SD(\text{Game } 11, \text{Game } 12) =$$
$$= \frac{1}{2}\left(1 - \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right]\right) + \frac{1}{2} \Pr\left[A\vec{d} = \vec{B} \text{ is not solvable}\right] =$$
$$= 1 - \Pr\left[A\vec{d} = \vec{B} \text{ is solvable}\right] \leq \frac{t}{(2^k)^{v-t+1}}.$$

Thus, the statistical distance between Game 10 and Game 12 is at most

$$\varepsilon_{mat} := \frac{t}{(2^k)^{v-t+1}}.$$

Now we can replace the linear equations in Game 9 with random $\vec{d}$ and get Game 13 that is $k \cdot \varepsilon_{mat}$ close to Game 9.

The same analysis can be done for finding $\vec{g}$ and thus we can replace it as well and get Game 14 that is ε_{mat} close to Game 13. We can also notice that Game $14 \equiv$ Game 3. This means we have found the computational distance between Game 3 and Game 2:

$$|\Pr[\text{Game } 2(\mathcal{A}) = 1] - \Pr[\text{Game } 3(\mathcal{A}) = 1]| \leq t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat}.$$

If $\mathcal{A}$ plays the role of P_1, then the proof is analogous. □

4.9 Completeness and security

Theorem 14. *MultiFunUSLESS is complete. That is, for all $\vec{a} \in (\{0,1\}^n)^t$, $\vec{b} \in \mathbb{F}_{2^k}^t$, all input $x \in \{0,1\}^n$ and all PRGs f, if φ is t-multi-point function defined by $\vec{a}$ and $\vec{b}$, then*

$$\Pr\left[\mathsf{Eval}^1(sk_1, x) + \mathsf{Eval}^2(sk_2, x) = \varphi(c) \mid sk_1, sk_2 \leftarrow \mathsf{Gen}(\vec{a}, \vec{b}, f)\right] = 1.$$

Note that this probability assumes Gen *did not abort.*

Proof. This follows straight from Theorems 11 and 12. □

Theorem 15. *MultiFunUSLESS is $(1, t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat})$-indistinguishable. That is, for all $\varepsilon_{\mathsf{PRG}}$-pseudo random generators f, if P_i is corrupted, $i \in \{1,2\}$, and P_{3-i} is not corrupted, then for all PPT algorithms $\mathcal{A}$*

$$\Pr\left[\textit{Game } 15(\mathcal{A}) = 1\right] - \frac{1}{2} \leq t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat},$$

where

$$\varepsilon_{mat} = \frac{t}{(2^k)^{v-t+1}}.$$

Proof. From Theorem 13 we know that MultiFunUSLESS is $(t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat})$-ROR indistinguishable. Let us assume towards contradiction that MultiFunUSLESS is not $(1, t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat})$-indistinguishable. Thus, there exists PPT algorithm $\mathcal{A}$ such that

$$\Pr\left[\text{Game } 15(\mathcal{A}) = 1\right] - \frac{1}{2} > t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat}.$$

Let us define a new adversary $\mathcal{B}$ against ROR indistinguishably in Algorithms 7 and 8:

```
Algorithm 7: $\mathcal{B}()$
1 $(a_1^0, b_1^0), \ldots, (a_t^0, b_t^0), (a_1^1, b_1^1), \ldots, (a_t^1, b_t^1), \sigma \leftarrow \mathcal{A}()$;
2 $b \xleftarrow{\$} \{0,1\}$;
3 return $(a_1^b, b_1^b), \ldots, (a_t^b, b_t^b)$;
```

```
Algorithm 8: $\mathcal{B}(sk_i)$
1 $\bar{b} \leftarrow \mathcal{A}(sk_i, \sigma)$;
2 return $[\bar{b} \overset{?}{=} b]$;
```

Without loss of generality, let us assume $i = 1$. Now let us inline $\vec{B}$ into Game 1 and 3. We get Game 16 and 17. In Game 17, we can delete line 3, because we do not use these values and move line 2 down. We get Game 18.

Let us calculate $|\Pr[\text{Game } 1(\mathcal{B}) = 1] - \Pr[\text{Game } 3(\mathcal{B}) = 1]|$. For this notice that Game $16 \equiv$ Game 15. Therefore,

$$\Pr[\text{Game } 1(\mathcal{B}) = 1] = \Pr[\text{Game } 16(\mathcal{B}) = 1] = \Pr[\text{Game } 15(\mathcal{A}) = 1] >$$
$$> \frac{1}{2} + t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat}.$$

Since in Game 18 b is sampled randomly just before the comparison with $\bar{b}$, then the probability that $b = \bar{b}$ must be $\frac{1}{2}$ i.e

$$\Pr[\text{Game } 3(\mathcal{B}) = 1] = \Pr[\text{Game } 18(\mathcal{B}) = 1] = \frac{1}{2}.$$

That is

$$|\Pr[\text{Game } 1(\mathcal{B}) = 1] - \Pr[\text{Game } 3(\mathcal{B}) = 1]| > t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat}.$$

We have reached a contradiction. Thus, MultiFunUSLESS is $(1, t \cdot n \cdot \varepsilon_{\mathsf{PRG}} + (n+1) \cdot \varepsilon_{mat})$-indistinguishable. $\square$

4.10 Efficiency

Let $t_{\mathbb{F}}$ denote the time for one field operation over $\mathbb{F}_{2^k}$, $t_{\mathbb{F}\,vec}$ the time for one operation over vector space $\mathbb{F}_{2^k}^v$ and t_{PRG} the time for one PRG evaluation.

Theorem 16. *Algorithm 1 produces keys with size $\Theta(vkn)$ bits.*

Proof. The key for P_i, $i \in \{1, 2\}$ is defined as ($[X_\epsilon]_i$, $[\tau_\epsilon]_i$, $\{w_{i,0}\}_{i=1}^n$, $\{w_{i,1}\}_{i=1}^n$, $\{\vec{d_i}\}_{i=0}^{n-1}$, $\vec{g}$). Element of $\mathbb{F}_{2^k}$ takes k bits to store. Let us analyze the key size element wise:

- $[X_\epsilon]_i \in \mathbb{F}_{2^k}^v$ takes $v \cdot k$ bits.
- $[\tau_\epsilon]_i \in \mathbb{F}_{2^k}$ takes k bits.
- $\{w_{i,0}\}_{i=1}^n \in \mathbb{F}_{2^k}^n$ takes $n \cdot k$ bits.
- $\{w_{i,1}\}_{i=1}^n \in \mathbb{F}_{2^k}^n$ takes $n \cdot k$ bits.
- $\{\vec{d_i}\}_{i=0}^{n-1} \in \mathbb{F}_{2^k}^{v \cdot n}$ takes $v \cdot n \cdot k$ bits.
- $\vec{g} \in \mathbb{F}_{2^k}^v$ takes $v \cdot k$ bits.

Let us sum this together and get

$$v \cdot k + k + n \cdot k + n \cdot k + v \cdot n \cdot k + v \cdot k = (v \cdot n + 2v + 2n + 1) \cdot k = \Theta(vkn).$$

$\square$

Lemma 2. *For* SubGen *to check, if a child node* $r||b$ *of an alive node* r *is supposed to be dead or alive, takes* $O(t\lceil\log t\rceil)$ *steps per level.*

Proof. Let us describe a possible algorithm for this operation.

1. For each alive node r keep a list of indices j for which $a_j^1||\dots||a_j^{|r|} = r$. Denote it by L_r.

2. During the check, iterate over L_r and check if $(|r|+1)$-th bit of a_j is equal to b. Return this truth value.

3. While iterating, divide indices into two lists $L_{r||0}$ and $L_{r||1}$.

Each index takes $\lceil\log t\rceil$ space and $\lceil\log t\rceil$ time steps to be read. On any given level there are t indices distributed among L_r and since only one bit of each a_j is looked at, the time complexity per level sums up to $O(t\lceil\log t\rceil)$. This means that when evaluating for all $r \in R_{i-1}$, then the total cost is $O(t\lceil\log t\rceil)$. □

Theorem 17. *Algorithm 2 (*SubGen*) is* $O(vt^2 \cdot t_{\mathbb{F}} + t \cdot t_{\mathbb{F}\,vec} + t \cdot t_{\mathsf{PRG}})$. *In fact it takes at most* $2t$ *PRG evaluations.*

Proof. Let us count the operations made by SubGen.

- Lines 1 to 5 take $\Theta(t_{\mathbb{F}})$ steps, because the operations are either over $\mathbb{F}_{2^k}$ or constant.

- Lines 6 to 13 take $O(t\lceil\log t\rceil)$, due to check on line 8 taking $O(t\lceil\log t\rceil)$ for all $r \in R_{i-1}$ and other operations can be done in constant time.

- Lines 14 to 25 take $O(t(t_{\mathbb{F}\,vec} + t_{\mathbb{F}}))$ steps, because there are at most t alive nodes in $R_i' \cup \widehat{R}_{i-1}$ and for each alive node a constant number of field and vector space operations are done.

- Line 26 takes $O(vt^2 \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec})$ steps. We can use Gaussian elimination on the system of linear equations, which takes $O(vt^2)$ field operations (see Theorem 20 in the Appendix). Choosing $\vec{d}_{i-1}$ is an operation in the vector space.

- Lines 27 to 32 take $O(t(2t_{\mathsf{PRG}} + t_{\mathbb{F}\,vec} + t_{\mathbb{F}}))$ steps. There are at most t alive nodes in R_i and for each alive node the PRG is called twice and some constant number of field operations and vector space operations are done.

Summing these together we get

$$\Theta(t_{\mathbb{F}}) + O(t\lceil\log t\rceil) + O(t(t_{\mathbb{F}\,vec} + t_{\mathbb{F}})) + O(vt^2 \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec}) + \\ + O(t(2t_{\mathsf{PRG}} + t_{\mathbb{F}\,vec} + t_{\mathbb{F}})) = O(vt^2 \cdot t_{\mathbb{F}} + t \cdot t_{\mathbb{F}\,vec} + t \cdot t_{\mathsf{PRG}}).$$

□

Theorem 18. *Algorithm 1 (*Gen*) is* $O(nvt^2 \cdot t_{\mathbb{F}} + nt \cdot t_{\mathsf{PRG}})$*. In fact it takes at most* $2tn$ *PRG evaluations.*

Proof. Let us count the operations made by Gen.

- Line 1 to 6 take $\Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}})$ steps, because there are constant number of field operations and vector space operations.
- Line 7 and 8 take $O(nt(vt \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec} + t_{\mathsf{PRG}}))$ steps, due to n evocations of SubGen.
- Line 9 and 10 take $\Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}})$ steps, because there are constant number of field operations and vector space operations.
- Line 11 takes $O(vt^2 \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec})$ steps. We can use Gaussian elimination on the system of linear equations, which takes $O(vt^2)$ field operations. Choosing $\vec{g}$ is an operation in the vector space.
- Lines 12 to 15 take $\Theta(vkn)$ steps, due to the key size.

Summing these together we get

$$\begin{aligned}\Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}}) + O(nt(vt \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec} + t_{\mathsf{PRG}})) + \Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}}) + \\ + O(vt^2 \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec}) + \Theta(vkn) = O(nt(vt \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec} + t_{\mathsf{PRG}}) + vkn).\end{aligned}$$

Let us assume that $t_{\mathbb{F}\,vec} \approx v \cdot t_{\mathbb{F}}$ and $t_{\mathbb{F}} \geq k$. Then

$$O(nt(vt \cdot t_{\mathbb{F}} + t_{\mathbb{F}\,vec} + t_{\mathsf{PRG}}) + vkn) = O(nvt^2 \cdot t_{\mathbb{F}} + nt \cdot t_{\mathsf{PRG}}).$$

□

Theorem 19. *Algorithm 3 (*Eval^P*) has time complexity* $\Theta(nvt_{\mathbb{F}} + nt_{\mathsf{PRG}})$*. In fact it takes* n *PRG evaluations.*

Proof. Let us count the operations made by Eval^P.

- Line 1 takes $\Theta(vkn)$ steps, due to the key size.
- Lines 2 to 4 take $\Theta(n(t_{\mathbb{F}\,vec} + t_{\mathbb{F}} + t_{\mathsf{PRG}}))$ steps, because lines 3 and 4 are repeated n times.
  - Lines 3 and 4 take $\Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}} + t_{\mathsf{PRG}})$ steps, because the scalar product is a vector space operation, addition and multiplication are field operations and the PRG is called once.
- Line 5 takes $\Theta(t_{\mathbb{F}\,vec} + t_{\mathbb{F}})$ steps, because of the scalar product and addition.

- Line 6 takes some constant number of steps.

Let us add these values together and get

$$\Theta(vkn) + \Theta(n(t_{\mathbb{F}vec} + t_{\mathbb{F}} + t_{\mathsf{PRG}})) + \Theta(t_{\mathbb{F}vec} + t_{\mathbb{F}}) + \Theta(1) = \\ = \Theta(n(t_{\mathbb{F}vec} + t_{\mathbb{F}} + t_{\mathsf{PRG}} + vk)).$$

Let us assume that $t_{\mathbb{F}vec} \approx v \cdot t_{\mathbb{F}}$ and $t_{\mathbb{F}} \geq k$. Then

$$\Theta(n(t_{\mathbb{F}vec} + t_{\mathbb{F}} + t_{\mathsf{PRG}} + vk)) = \Theta(n(v \cdot t_{\mathbb{F}} + t_{\mathbb{F}} + t_{\mathsf{PRG}} + vk)) = \Theta(nvt_{\mathbb{F}} + nt_{\mathsf{PRG}}).$$

□

4.11 Games

```
Game 1:
  Input: $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $sk_1 \overset{P_1}{\leftarrow} \mathsf{Gen}((a_1, b_1), \ldots, (a_t, b_t), f)$;
3 return $\mathcal{A}(sk_1)$;
```

```
Game 2:
  Input: $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $sk_2 \overset{P_2}{\leftarrow} \mathsf{Gen}((a_1, b_1), \ldots, (a_t, b_t), f)$;
3 return $\mathcal{A}(sk_2)$;
```

```
Game 3:
  Input: $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $X_\epsilon \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
3 $\tau_\epsilon \overset{\$}{\leftarrow} \mathbb{F}_{2^k}$;
4 for $i = 1$ to $n$ do
5     $w_{i,0} \overset{\$}{\leftarrow} \mathbb{F}_{2^k} \setminus \{0\}$;
6     $w_{i,1} \overset{\$}{\leftarrow} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
7     $\vec{d}_{i-1} \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
8 $\vec{g} \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
9 $sk \leftarrow X_\epsilon, \tau_\epsilon, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g}$;
10 return $\mathcal{A}(sk)$;
```

Game 4:

```
Input : $\mathcal{B}$ - the adversarial algorithm
1 $x \xleftarrow{\$} \mathbb{F}_{2^k}$;
2 $y \leftarrow f(x)$;
3 return $\mathcal{B}(y)$;
```

Game 5:

```
Input : $\mathcal{B}$ - the adversarial algorithm
1 $y \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
2 return $\mathcal{B}(y)$;
```

Game 6:

```
Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $\cdots$;
3 for $r||b \in R_i$ do
4     $([X_{r||b}]_1, [\tau_{r||b}]_1) \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
5     $([X_{r||b}]_2, [\tau_{r||b}]_2) \leftarrow f(\left\langle [X_r]_2, \vec{d}_{i-1} \right\rangle + [\tau_r]_2 \cdot w_{i,b})$;
6     $X_{r||b} \leftarrow [X_{r||b}]_1 + [X_{r||b}]_2$;
7     $\tau_{r||b} \leftarrow [\tau_{r||b}]_1 + [\tau_{r||b}]_2$;
8 $\cdots$;
9 return $\mathcal{A}(sk_2)$;
```

Game 7:

```
1 $\cdots$;
2 for $r||b \in R_i$ do
3     $(X_{r||b}, \tau_{r||b}) \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
4 $\cdots$;
```

Game 8:

```
Input : $\mathcal{A}$ - the adversarial algorithm
$(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
$[X_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
$[\tau_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}$;
$R_0 \leftarrow \{\epsilon\}$, where $\epsilon$ is an empty string;
for $i = 1$ to $n$ do
    $\cdots$;
    Let $A$ be a $|R_{i-1}| \times v$ matrix of zeroes;
    Let $\vec{B}$ be a zero column vector of length $|R_{i-1}|$;
    $j_i \leftarrow 1$;
    for $r||b \in R'_i$ do
        $(X_r, \tau_r) \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
        Set the $j_i$-th row of $A$ to be $X_r$;
        Set the $j_i$-th element of $\vec{B}$ to be $\tau_r \cdot w_{i,b}$;
        $j_i \leftarrow j_i + 1$ ;
    for $r \in \widehat{R}_{i-1}$ do
        $w_{r,2} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{w_{i,0}, w_{i,1}\}$;
        $(X_r, \tau_r) \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
        Set the $j_i$-th row of $A$ to be $X_r$;
        Set the $j_i$-th element of $\vec{B}$ to be $\tau_r \cdot w_{r,2}$;
        $j_i \leftarrow j_i + 1$ ;
    Solve $A\vec{d}_{i-1} = \vec{B}$ and sample $\vec{d}_{i-1}$ from the solution space;
for $i = 1$ to $t$ do
    $(X_{a_i}, \tau_{a_i}) \xleftarrow{\$} \mathbb{F}_{2^k}^{v+1}$;
$A \leftarrow \begin{pmatrix} X_{a_1} \\ \vdots \\ X_{a_t} \end{pmatrix}$;
$\vec{B} \leftarrow (b_1 + \tau_{a_1}, \cdots, b_t + \tau_{a_t})^T$;
Solve $A\vec{g} = \vec{B}$ and sample $\vec{g}$ from the solution space;
return $\mathcal{A}([X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$;
```

```
Game 9:
  Input: $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $[X_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
3 $[\tau_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}$;
4 $R_0 \leftarrow \{\epsilon\}$, where $\epsilon$ is an empty string;
5 for $i = 1$ to $n$ do
6     $w_{i,0} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0\}$;
7     $w_{i,1} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
8     $R_i \leftarrow \emptyset$;
9     $R'_i \leftarrow \emptyset$;
10    $\widehat{R}_{i-1} \leftarrow \emptyset$;
11    for $r \in R_{i-1}$ do
12        for $b \in \{0, 1\}$ do
13            if $\exists a_j$ such that $a_j$ starts with $r||b$ then
14                $R_i \leftarrow R_i \cup \{r||b\}$;
15            else
16                $R'_i \leftarrow R'_i \cup \{r||b\}$;
17        if $r||0 \in R_i$ and $r||1 \in R_i$ then
18            $\widehat{R}_{i-1} \leftarrow \widehat{R}_{i-1} \cup \{r\}$;
19    $A \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}| \times v}$;
20    $\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}|}$;
21    Solve $A\vec{d}_{i-1} = \vec{B}$ and sample $\vec{d}_{i-1}$ from the solution space;
22 $A \xleftarrow{\$} \mathbb{F}_{2^k}^{t \times v}$;
23 $\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{t}$;
24 Solve $A\vec{g} = \vec{B}$ and sample $\vec{g}$ from the solution space;
25 return $\mathcal{A}([X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$;
```

```
Game 10:
1 $A \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}| \times v}$;
2 $\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}|}$;
3 Solve $A\vec{d} = \vec{B}$ and choose a $\vec{d}$;
4 return $\vec{d}$;
```

Game 11:

```
1  $A \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}| \times v}$;
2  $\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{|R_{i-1}|}$;
3  if $\mathsf{rank}(A) < \mathsf{rank}(A|\vec{B})$ then
4      return $\perp$;
5  Solve $A\vec{d} = \vec{B}$ and choose a $\vec{d}$;
6  return $\vec{d}$;
```

Game 12:

```
1  $\vec{d} \xleftarrow{\$} \mathbb{F}_{2^k}^{v}$;
2  return $\vec{d}$;
```

Game 13:

```
   Input : $\mathcal{A}$ - the adversarial algorithm
1  $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2  $[X_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}^{v}$;
3  $[\tau_\epsilon]_2 \xleftarrow{\$} \mathbb{F}_{2^k}$;
4  for $i = 1$ to $n$ do
5      $w_{i,0} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0\}$;
6      $w_{i,1} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
7      $\vec{d}_{i-1} \xleftarrow{\$} \mathbb{F}_{2^k}^{v}$;
8  $A \xleftarrow{\$} \mathbb{F}_{2^k}^{t \times v}$;
9  $\vec{B} \xleftarrow{\$} \mathbb{F}_{2^k}^{t}$;
10 Solve $A\vec{g} = \vec{B}$ and sample $\vec{g}$ from the solution space;
11 return $\mathcal{A}([X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^{n}, \{w_{i,1}\}_{i=1}^{n}, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$;
```

Game 14:

```
Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow \mathcal{A}()$;
2 $[X_\epsilon]_2 \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
3 $[\tau_\epsilon]_2 \overset{\$}{\leftarrow} \mathbb{F}_{2^k}$;
4 for $i = 1$ to $n$ do
5     $w_{i,0} \overset{\$}{\leftarrow} \mathbb{F}_{2^k} \setminus \{0\}$;
6     $w_{i,1} \overset{\$}{\leftarrow} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
7     $\vec{d}_{i-1} \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
8 $\vec{g} \overset{\$}{\leftarrow} \mathbb{F}_{2^k}^v$;
9 return $\mathcal{A}([X_\epsilon]_2, [\tau_\epsilon]_2, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g})$;
```

Game 15:

```
Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1^0, b_1^0), \ldots, (a_t^0, b_t^0), (a_1^1, b_1^1), \ldots, (a_t^1, b_t^1), \sigma \leftarrow \mathcal{A}()$;
2 $b \overset{\$}{\leftarrow} \{0, 1\}$;
3 $sk_1, sk_2 \leftarrow \mathsf{Gen}((a_1^b, b_1^b), \ldots, (a_t^b, b_t^b), f)$;
4 $\bar{b} \leftarrow \mathcal{A}(sk_i, \sigma)$;
5 return $[\bar{b} \overset{?}{=} b]$;
```

Game 16:

```
Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1^0, b_1^0), \ldots, (a_t^0, b_t^0), (a_1^1, b_1^1), \ldots, (a_t^1, b_t^1), \sigma \leftarrow \mathcal{A}()$;
2 $b \overset{\$}{\leftarrow} \{0, 1\}$;
3 $sk_1 \overset{P_1}{\leftarrow} \mathsf{Gen}((a_1^b, b_1^b), \ldots, (a_t^b, b_t^b), f)$;
4 $\bar{b} \leftarrow \mathcal{A}(sk_1, \sigma)$;
5 return $[\bar{b} \overset{?}{=} b]$;
```

```
Game 17:
  Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1^0, b_1^0), \ldots, (a_t^0, b_t^0), (a_1^1, b_1^1), \ldots, (a_t^1, b_t^1), \sigma \leftarrow \mathcal{A}()$;
2 $b \xleftarrow{\$} \{0, 1\}$;
3 $(a_1, b_1), \ldots, (a_t, b_t) \leftarrow (a_1^b, b_1^b), \ldots, (a_t^b, b_t^b)$;
4 $X_\epsilon \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
5 $\tau_\epsilon \xleftarrow{\$} \mathbb{F}_{2^k}$;
6 for $i = 1$ to $n$ do
7     $w_{i,0} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0\}$;
8     $w_{i,1} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
9     $\vec{d}_{i-1} \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
10 $\vec{g} \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
11 $sk \leftarrow X_\epsilon, \tau_\epsilon, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g}$;
12 $\bar{b} \leftarrow \mathcal{A}(sk, \sigma)$;
13 return $[\bar{b} \overset{?}{=} b]$;
```

```
Game 18:
  Input : $\mathcal{A}$ - the adversarial algorithm
1 $(a_1^0, b_1^0), \ldots, (a_t^0, b_t^0), (a_1^1, b_1^1), \ldots, (a_t^1, b_t^1), \sigma \leftarrow \mathcal{A}()$;
2 $X_\epsilon \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
3 $\tau_\epsilon \xleftarrow{\$} \mathbb{F}_{2^k}$;
4 for $i = 1$ to $n$ do
5     $w_{i,0} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0\}$;
6     $w_{i,1} \xleftarrow{\$} \mathbb{F}_{2^k} \setminus \{0, w_{i,0}\}$;
7     $\vec{d}_{i-1} \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
8 $\vec{g} \xleftarrow{\$} \mathbb{F}_{2^k}^v$;
9 $sk \leftarrow X_\epsilon, \tau_\epsilon, \{w_{i,0}\}_{i=1}^n, \{w_{i,1}\}_{i=1}^n, \{\vec{d}_i\}_{i=0}^{n-1}, \vec{g}$;
10 $\bar{b} \leftarrow \mathcal{A}(sk, \sigma)$;
11 $b \xleftarrow{\$} \{0, 1\}$;
12 return $[\bar{b} \overset{?}{=} b]$;
```

5 Comparison to prior works

The study of function secret sharing schemes dates back to 2014 with [GI14] and 2015 with [BGI15], [BGI16]. These research papers focused on point functions and the general topic of function secret sharing. In the following years, applications for FSS schemes required the use of multi-point functions instead of point functions. The following two approaches were developed in [BCGI18]:

- The most trivial approach of getting efficient multi-point function is to repeat the DPF scheme multiple times. We call this approach the simple scheme. For a t-multi-point function the key size is t times bigger than in the point function case and both evaluation and generation algorithms take t times longer.
- The second kind of approach is using combinatorial batch codes to distribute the information into m buckets and use the DPF construction on these smaller batches of information to make it secure. The drawback is that cheap evaluation of a single value becomes impossible, because this approach optimises for the full evaluation of the domain. However, full evaluation becomes a lot cheaper. We call this approach the batch code scheme.

In Table 1 we can see how the three schemes compare to each other. Parameters t, n, k come from that t-multi-point functions are taken from the function family $\mathcal{F}(\{0,1\}^n \to \mathbb{F}_{2^k})$, $\epsilon > 0$ is a constant used in the batch-code scheme and v is the security parameter.

Scheme	Key size	Number of PRG calls in Gen	Number of PRG calls in Eval
Simple scheme	$O(t(vn + k))$	$O(tn)$	tn
Batch-code scheme	$O(t^{1+\epsilon}(v\lceil n - \log t\rceil + k))$	$O(n)$	$O(n)$
MultiFunUSLESS	$O(vnk)$	$O(tn)$	n

Table 1. Multi-point function secret sharing scheme comparison

Since other schemes work over bits and in the end convert to $\mathbb{F}_{2^k}$ and MultiFunUSLESS works over $\mathbb{F}_{2^k}$ the entire time, then the key size can be quite large for MultiFunUSLESS. If t is bigger than k, then the key becomes of the same order as in the other schemes.

Number of PRG calls in the generation algorithm is the best for the batch-code scheme and if we go into the details of it, the second place belongs to MultiFunUSLESS, because our scheme does less or equal to $2tn$ calls and the simple scheme does exactly $2tn$ calls. On the other hand, MultiFunUSLESS has to solve $n + 1$ systems of linear equations over the finite field $\mathbb{F}_{2^k}$. Thus, another costly operation might have been introduced and the win in PRG calls might be moot. The best score in the efficiency category for MultiFunUSLESS is the number of PRG calls in the evaluation algorithm. It does t times less calls to the PRG than the simple algorithm and some constant number times less calls as the batch-code scheme. The batch-code scheme does more PRG calls, due to having to do the full evaluation of the domain and then finding the evaluation of a single point. This is why MultiFunUSLESS outperforms batch-code approach, when we do some small number of single evaluations of φ.

Overall we can see that our scheme is the best choice if $t \geq k$ and we want to do some constant number of evocations of Eval. One other bonus is that the other schemes need one PRG and one map that maps random elements of $\{0,1\}^v$ to $\mathbb{F}_{2^k}$, but our scheme only needs the PRG.

6 Conclusions and future work

We presented a new multi-point function secret sharing scheme MultiFunUSLESS and proved that it is complete and secure with a generation function that passes with high probability. The scheme has the biggest advantage over its competitors if just a small number of single evaluations are needed. Moreover, it is built from simple mathematical constructions and it does not require two pseudorandom primitives as the other construction do. But there is room for optimization and improvement, because in many cases the key size is much bigger. In the future, we could analyse, if it is necessary to work over $\mathbb{F}_{2^k}$, or could we work over some smaller field, which would make the algorithm more efficient. We could try to pack the key to make it smaller as it is done in [BGI16]. On the other side, we could look at the possibility of generalizing the finite field $\mathbb{F}_{2^k}$ setting to the ring $\mathbb{Z}_{2^k}$ setting to further boost the efficiency as arithmetic over the ring is faster than arithmetic over the field. On the more practical side, the scheme should be implemented in real life and then performance against other multi-point FSS schemes should be measured to see the effects of solving systems of linear equations on the time of the generation algorithm. Furthermore, since Gen needs a trusted setup to work, a line of inquiry would be to change it to an interactive protocol between the parties that can work without the need to trust a third party.

In conclusion, MultiFunUSLESS has much potential in becoming even faster and more efficient. In this thesis the first big steps were taken towards this.

References

[AR91] Howard Anton and Chris Rorres. *Elementary linear algebra: applications version*. John Wiley & Sons, sixth edition, 1991.

[BCG$^+$19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent ot extension and more. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*, pages 489–518. Springer, 2019.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector ole. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 896–912, 2018.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[BGI15] 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.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, 2016.

[Car00] Neal L Carothers. *Real analysis*. Cambridge University Press, 2000.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, pages 1–20. Springer, 2008.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 640–658. Springer, 2014.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge university press, 2001.

[Gra72] Mary W Gray. *Calculus with finite mathematics for social sciences*. Addison-Wesley Publishing Company, Inc., 1972.

[GW09] Geoffrey Grimmett and Dominic Welsh. *Probability: an introduction*. Oxford University Press, 2009.

[IEE] What is multiparty computation? `https://digitalprivacy.ieee.org/publications/topics/what-is-multiparty-computation`. Accessed on 14.05.2024.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and hall/CRC, 2008.

[KL23] Stephan Krenn and Thomas Lorünser. *An Introduction to Secret Sharing: A Systematic Overview and Guide for Protocol Selection*. Springer Nature, 2023.

[MP13] Gary L Mullen and Daniel Panario. *Handbook of finite fields*, volume 17. CRC press Boca Raton, 2013.

[MSS08] Kurt Mehlhorn, Peter Sanders, and Peter Sanders. *Algorithms and data structures: The basic toolbox*, volume 55. Springer, 2008.

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

Appendix

I. Additional proofs

Time complexity of Gaussian elimination

Theorem 20. *Let* $t, v, k \in \mathbb{N}$*,* $v > t$*,* $A \in \mathbb{F}_{2^k}^{t\times v}$ *and* $\vec{B} \in \mathbb{F}_{2^k}^{t}$*. Gaussian elimination on the matrix equation* $Ax = \vec{B}$ *takes* $O(vt^2)$*.*

Proof. Let us describe Gaussian elimination [AR91] line by line and bound the time complexity of each step. Operations on rows are done on the augmented matrix $(A \mid \vec{B})$.

Start iterating over columns of A from left to right. Let j be the column index. And S be an empty set at the beginning.

1. Search for a row $i \notin S$ with non-zero element in j-th column. It takes at most t field operations.

2. If all elements were 0, then move onto the next column and go back to the previous step.

3. If a non-zero element a was found, find its inverse a^{-1} and add i to set S. This takes one field operation and some constant number of steps.

4. Multiply the elements of this row i with a^{-1} and replace the i-th row with these values. This takes $O(v)$ field operations.

5. For each row $i' \notin S$ take the j-th element b. Multiply each element of the i-th row with b and subtract these values from the elements of the i'-th row and replace the i'-th row with the result. This takes $O(tv)$ field operations.

6. Repeat for column $j + 1$ until $j = v$ or $|S| = t$.

After this the matrix is in the row-echelon form. The worst case scenario is if first $v - t$ columns are all zero columns and the last t are linearly independent of each other. In this case, we do $O(t(v - t)) = O(tv)$ field operations to check the zero columns and $O(t(t + 1 + tv + 1)) = O(vt^2)$ field operations to get the row-echelon form.

To get to the reduced row-echelon form, we do similar triangular process, but look at column indices in S. This is similar enough that it also takes $O(vt^2)$ field operations. □

II. Licence