Reducing the Share Size of Weighted Threshold Secret Sharing Schemes via Chow Parameters Approximation

Oriol Farràs and Miquel Guiot

Universitat Rovira i Virgili, Tarragona, Spain
oriol.farras@urv.cat, miquel.guiot@urv.cat

Abstract. A secret sharing scheme is a cryptographic primitive that allows a dealer to share a secret among a set of parties, so that only authorized subsets of them can recover it. The access structure of the scheme is the family of authorized subsets.
In a weighted threshold access structure, each party is assigned a weight according to its importance, and the authorized subsets are those in which the sum of their weights is at least the threshold value. For these access structures, the share size of the best known secret sharing schemes is either linear on the weights or quasipolynomial on the number of parties, which leads to long shares, in general.
In certain settings, a way to circumvent this efficiency problem is to approximate the access structure by another one that admits more efficient schemes. This work is dedicated to the open problem posed by this strategy: Finding secret sharing schemes with a good tradeoff between the efficiency and the accuracy of the approximation.
We present a method to approximate weighted threshold access structures by others that admit schemes with small shares. This method is based on the techniques for the approximation of the Chow parameters developed by De et al. [Journal of the ACM, 2014]. Our method provides secret sharing schemes with share size $n^{1+o(1)}$, where n is the number of parties, and whose access structure is *close* to the original one. Namely, in this approximation the condition of being authorized or not is preserved for almost all subsets of parties.
In addition, applying the recent results on computational secret sharing schemes by Applebaum et al. [STOC, 2023] we show that there exist computational secret sharing schemes whose security is based on the RSA assumption and whose share size is polylogarithmic in the number of parties.

Keywords: Secret sharing scheme · weighted threshold access structure · threshold cryptography · Chow parameters

The authors are supported by grant 2021 SGR 00115 from the Government of Catalonia and by the project HERMES, funded by INCIBE and by the European Union NextGeneration EU/PRTR, and the project ACITHEC PID2021-124928NB-I00, funded by MCIN/AEI/10.13039/501100011033/FEDER, EU.

1 Introduction

A secret sharing scheme is a cryptographic primitive that allows a dealer to share a secret among a set of parties, so that only authorized subsets of them can recover it. These authorized subsets define the access structure of the scheme, which is a monotone increasing family of parties.

Secret sharing schemes were introduced independently by Shamir [Sha79] and Blakley [Bla79] in 1979, when they presented methods for constructing secret sharing schemes for threshold access structures. In these schemes, the authorized subsets are those containing at least a given threshold of parties. Interestingly, these constructions are ideal, in the sense that the length of the share received by each party is equal to the length of the secret. This is the best situation we can hope for [KGH83].

Secret sharing schemes are used as a building box in cryptographic protocols such as multiparty computation and threshold encryption, for example [Bei11]. In many of these applications, schemes with threshold access structures are enough. However, there are situations that require non-threshold access structures. This is the case when using secret sharing schemes for protocols in the proof-of-stake model, where each validator has a stake that depends on the amount of coins it has and the importance in the system is proportional to the stake, resulting in a non-uniformly distribution [KRDO17,BCC$^+$21,DPTX24]. Another common case can be found in the stock exchange, where the shares of a company are non-uniformly distributed among the shareholders and the weight of their vote depends on the share. In such cases, there is a need of secret sharing schemes with *weighted threshold access structures* (WTASs). In these access structures, each party is assigned a weight according to its importance and the authorized subsets are those in which the sum of their weights is at least the threshold value.

The share size of best known secret sharing schemes for general weighted threshold access structures is either linear on the weights [Sha79] or quasipolynomial on the number of parties [BW06], which may lead to long shares, in general. Recently, there were proposals trying to circumvent this efficiency problem by approximating the weights by smaller ones or by relaxing the privacy and correctness problem of the access structure [BHS23,GJM$^+$23,DPTX24], considering approximations of weighted threshold access structures. This work is dedicated to the open problem posed by this strategy: Finding secret sharing schemes with a good tradeoff between the efficiency and the accuracy of the approximation.

1.1 Our Results

In this work, we present a method that, given a weighted threshold access structure Γ, it provides a secret sharing scheme with *small* share size that realizes a weighted threshold access structure Γ' that is *close* to Γ. For that, we translate our problem into a problem of approximation of monotone Linear Threshold Functions (LTF) and, in this richer complexity theory framework, we develop techniques for the approximation of these functions. As a measure of distance

between access structures, we consider the Hamming distance. Namely, the error of the approximation is $d/2^n$, where d is the distance between the access structures and n is the number of parties of the access structure. Our main result is the following.

Theorem 1.1 (Informal). *For any weighted threshold access structure Γ on n parties there exists a secret sharing scheme with share size $n^{1+o(1)}$ whose access structure is $o(1)$-close to Γ.*

The result is constructive, and we provide an efficient algorithm that, given a weighted threshold access structure Γ, it outputs another weighted threshold access structure Γ' whose weights are much smaller, the distance between Γ and Γ' is small, and the hierarchy among parties is preserved. Then, with the new weights, it is enough to use the construction of Shamir [Sha79] for weighted threshold access structures to obtain the secret sharing scheme. The approximation error is $o(1)$, which means that for almost all subsets of parties the condition of being authorized or not is not modified in the approximation. Moreover, our scheme is linear for finite fields $\mathbb{F}$ with size $\log |\mathbb{F}| = \Omega(\log n)$.

Previous best solutions had different trade-offs, illustrated in Fig. 1. Our result is in the last row. In some previous works, the bounds are for the total share size, and not for the share size. Because of that, we decided to present the bounds for the total share size, that is, the sum of the size of the shares of all parties. In our construction, it is simply $n \cdot n^{1+o(1)}$.

Our main technical contribution is the use of Chow parameters in the construction of secret sharing schemes for weighted threshold access structures. In this regard, the approximation of linear threshold functions by Chow parameters is a problem that has been thoroughly studied in the past, giving positive and negative results (see, for example, [Ser06,OS11,DDFS14]). In particular, we modify an algorithm for the approximation of Chow parameters by De et al. [DDFS14] to solve this problem in the monotone case. That is, to guarantee that all the weights of the approximation are positive. With that, we can approximate weighted threshold access structures with small weights. In this context, our main result is the following.

Theorem 1.2 (Informal). *Let f be a monotone LTF. For any $0 < \epsilon$ there exists an ϵ-close monotone LTF g represented by an integer vector with norm $\sqrt{n} \cdot \text{quasipoly}\left(\frac{1}{\epsilon}\right)$.*

Previous proposals also consider reductions of the weights, by scaling or rounding them [BHS23,DPTX24], leading to approximated access structures. These techniques can be replaced by the the Chow parameters approximation technique developed in this work. Furthermore, we give a lower bound on the size of the weights obtained by any approximation technique and show that our strategy is nearly optimal for this task. This is summarized in the following statement.

Theorem 1.3 (Informal). *Let $\epsilon \in (0, 1)$, and let $n \in \mathbb{N}$. There exists a monotone LTF f over n variables such that any monotone LTF ϵ-close to f has integer weights of size $\Omega\left(\sqrt{n}, \text{quasipoly}\left(\frac{1}{\epsilon}\right)\right)$.*

	Total Share Size	**Access structure**	**Error**	**Privacy**	**Linear**
[Sha79]	$W \log n = 2^{O(n \log(n))}$	WTAS	0	Perfect	Yes
[BW06]	$n^{O(\log(n))}$	WTAS	0	Perfect	Yes
[GJM+23]	$W = 2^{O(n \log(n))}$	$(t, t + \Omega(\lambda))$-ramp WTAS	-	$2^{-\lambda}$-Stat.	No
[BHS23] Rounding	$O\left(\frac{n}{\beta - \alpha}\right)$	$(\alpha W, \beta W)-$ramp WTAS	-	Perfect	Yes
[BHS23] BS Channels	$n \cdot \max\left\{\lambda^2, \text{poly}(\frac{1}{\alpha - \beta})\right\}$	$(\alpha W, \beta W)-$ramp WTAS	-	$2^{-\lambda}$-Stat.	No
Theorem 1.1	$n^{2+o(1)}$	WTAS	$o(1)$	Perfect	Yes

Fig. 1. Summary of secret sharing schemes for weighted threshold access structures with information-theoretic security. In the second column, we give an upper bound on the total share size of the schemes, considering secrets of 1-bit. We use the fact that the best upper bound for the size of the weights in weighted threshold access structures is $W = 2^{O(n \log n)}$ [Mur71], which is tight [Hås94]. In the last two rows, the bound is obtained by multiplying the bound on the individual share size by n. In the *Access structure* column, we distinguish between perfect WTAS and ramp WTAS. In the *Error* column, we bound the error in the approximation of weighted threshold access structures. If the scheme realize exactly the access structure, we set the error to be 0. Ramp WTASs can also be used to approximate perfect WTAS, but we could not find any non-trivial upper bound on the error. In the *Privacy* column, we distinguish between perfect and statistical privacy. In the last column, we show if the scheme is linear or not.

If, instead of constructing schemes in the information-theoretic setting, we consider computational assumptions, it is possible to build more efficient secret sharing schemes. This can be done by applying the recent results on succinct computational secret sharing schemes by Applebaum et al. [ABI+23] to the polynomial size monotone circuits for weighted threshold functions constructed by Beimel and Weinreb [BW06]. More in detail, the existence of an efficient Projective Pseudorandom Generator (pPRG) allows to obtain a computational secret sharing scheme for weighted threshold access structures with polylogarithmic share size in the number of parties. This is stated in the following theorem.

Theorem 1.4 (Informal). *Under the subexponential RSA assumption, any weighted threshold access structure over n parties admits a computational secret sharing scheme where the size of the shares is $O(\text{polylog}(n))$ and the size of the public information is $O(\text{poly}(n))$.*

Furthermore, we show how to improve the result of Theorem 1.4 by combining it with our approximation technique. In particular, we specify the public information size (the degree of its polynomial) at the cost of considering an $o(1)$-close weighted threshold access structure.

Open Questions. Our results leave several open questions about the search of better secret sharing schemes for weighted threshold access structures. The schemes presented in this work can be improved in the two main stages, which are the approximation of the weighted threshold access structure with small weights and the construction of schemes for small weights. The approximation techniques in the first stage have limitations that are similar to the ones in [DDFS14], which are close to the optimal, in the worst case. Any improvement in the average case of this technique would lead to smaller shares in the average case.

In the second stage, our approach is simpler, because we directly apply existing techniques by Shamir [Sha79] and Applebaum et al. [ABI+23] that take benefit of a short description of the access structure by means of weights or formulas. It is natural to expect improvements in this stage by using more complex formulas with more complex gates [LV18] or with wiretap techniques [BHS22].

Improving the upper and lower bounds on the share size for weighted threshold access is still the most important open problem in this area. Also, we do not have the intuition of what makes an access structure *hard* to be realized. On the positive side, we have characterizations of ideal weighted threshold access structures [BW06,FP12]. However, we do not know how to take advantage of this to find a useful characterization of access structures that admit schemes with polynomial share size.

1.2 Our Techniques

Given a weighted threshold access structure Γ, our main objective is to construct another weighted threshold access structure Γ' that is close to Γ and whose weights are small. In this way, the impact of the weights on the share size of any secret sharing scheme realizing Γ' will be considerably low.

With this in mind, since every access structure can be described by a monotone Boolean function, the starting point of our work is to translate the problem of approximating weighted threshold access structures to the problem of approximating monotone Boolean functions.

In the case of a weighted threshold access structure defined by a threshold T and a vector of positive weights $\boldsymbol{w} = (w_1, \ldots, w_n)$ assigned to the parties, it can be represented by a monotone Boolean function of the form

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} - T),$$

which are known as monotone LTFs in the context of complexity theory.

We can therefore study weighted threshold access structures from the perspective of complexity theory simply by considering the monotone LTF assigned to it. More in detail, our proposal consists in approximating Γ by reducing the weights and the threshold of its associated monotone LTF f. We do this by taking advantage of the description of f in terms of its Chow parameters. This procedure is summarized in Fig. 2 and can be done in five steps.

1. Consider the monotone LTF f associated to Γ.

2. Compute the Chow parameters $\boldsymbol{\chi}_f$ of f.
3. Modify the Chow parameters $\boldsymbol{\chi}_f$ to obtain the Chow parameters $\boldsymbol{\chi}_g$ of a monotone LTF g that is ϵ-close to f and has small weights.[1]
4. Construct the monotone LTF g from $\boldsymbol{\chi}_g$.
5. Consider the weighted threshold access structure Γ' associated to g.

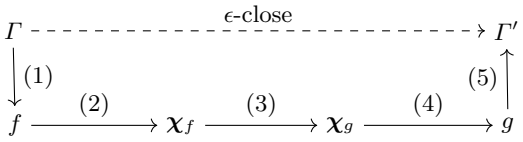

Fig. 2. Procedure for approximating any weighted threshold access structure.

Steps (1), (2), and (5) of Fig. 2 are immediate. For this reason, throughout this work we skip steps (1) and (5) and we deal directly with the monotone LTFs associated to the access structures. All the effort remains in deriving steps (3) and (4). To do so, we adapt the results of De et al. [DDFS14], in which they construct an approximate LTF with smaller weights by solving a problem related to the Chow parameters, described next.

The Chow Parameters Problem. Any Boolean function can be uniquely expressed as a real multilinear polynomial [O'D14], whose degree-0 and degree-1 coefficients are known as the Chow parameters.[2] The notion of the Chow parameters is of greater importance in the case of LTFs, since they uniquely specify any LTF within the space of all Boolean functions [Cho61] and, when the function is monotone, they quantify the influence of each variable on the output result. In this context, the Chow parameters problem consists in efficiently reconstructing any LTF from its Chow parameters.

In their work, De et al. [DDFS14] solve the approximate version of the Chow parameters problem by constructing a LTF close to the original one based on its Chow parameters. More specifically, they they design an iterative algorithm that starts with an approximation of the Chow parameters and step by step modifies them until the desired LTF is obtained. Moreover, their construction has the advantage that the resulting LTF has weights that depend sublinearly in the input length and quasipolynomially in the error.

In this work, we adapt the construction of De et al. [DDFS14] to the monotone setup to guarantee that the resulting approximation not only is a LTF but also a monotone LTF. Moreover, we also analyse the running time of the procedure to ensure its efficiency.

[1] The notion of distance between Boolean functions consists in the fraction of inputs in which they differ.

[2] Indeed, the Chow parameters are a particular case of a much more general family known as the Fourier coefficients.

Secret Sharing Schemes Construction. Once we have derived a low-weight approximator for the original access structure, we still need to construct the information-theoretic and the computational secret sharing schemes. In both cases, we apply already known constructions that allow us to maximize the benefits of having an approximate weighted threshold access structure with small weights.

In the case of the information-theoretic scheme, we use Shamir's virtualization technique [Sha79]. In this way, the share size of the resulting scheme is equal to the total weight, which we know in advance that is small due to the approximation procedure.

With respect to the computational setting, we apply a recent result of Applebaum et al. [ABI+23] in which they introduce a new cryptographic primitive known as Projective Pseudorandom Generator (pPRG). They use it to obtain secret sharing schemes for monotone circuits in which the size of the shares is polylogarithmic in the number of gates. More in detail, we combine this construction with the existence of a monotone circuit of polynomial size for any weighted threshold access structure [BW06] to obtain a scheme with polylogarithmic shares. Furthermore, we use our approximation technique to bound the size of the public information.

In both cases we also tune the parameters in order to find a good trade-off between the accuracy of the approximation and the share size. In particular, we perform a fine-grained analysis of the size of the Chow parameters and use their interpretation as the influence of the parties in the access structure to obtain secret sharing schemes with even smaller share size.

1.3 Related Work

In this section, we restrict the discussion to the previous works on secret sharing schemes for weighted threshold access structures.

In 1979, Shamir [Sha79] and Blakley [Bla79] presented the first secret sharing schemes for threshold access structures. Shamir also presented a way to realize weighted threshold access structures with threshold schemes via *virtualization*. Given the weights w_i and the threshold t, the dealer treats party i as w_i different parties, sending w_i different shares of the t-threshold scheme to party i. This technique gives a scheme with total share size $O(W \log W)$ for any access structure, where W is the sum of the weights.

Weighted threshold access structures are a specific kind of *hierarchical access structures*. In these access structures, parties are partitioned into clusters that are hierarchically ordered, being the parties in higher levels more powerful than the ones in lower levels. Simmons [Sim88] considered some hierarchical access structures, and Brickell [Bri89] found ideal linear schemes for them, providing new tools for the construction of schemes with linear properties. By using different kinds of polynomial interpolation, Tassa [Tas07], and Tassa and Dyn [TD09] proposed constructions of ideal secret sharing schemes for some kinds of hierarchies. Beimel, Tassa and Weinreb [BTW08] presented a characterization of the ideal weighted threshold access structures, generalizing some

partial results in [MPSV99,PS00]. Farràs and Padró [FP12] characterized all ideal hierarchical access structures, and presented ideal linear schemes for them. That work included an alternative characterization of ideal weighted threshold access structures. Characterizations in [BTW08,FP12] use the connections between ideal access structures and matroids by Brickell and Davenport [BD91]. Indeed, Mo [Mo23] found that ideal hierarchical access structures are connected to lattice path matroids. Beyond the ideal case, the characterization of weighted threshold access structures that admit efficient schemes is open.

Beimel and Weinreb [BW06] proved that all weighted threshold access structures admit secret sharing schemes in which the size of the shares is $n^{O(\log(n))}$. Taking into account that most weighted threshold access structures require weights of exponential size [SB91], this is the best general construction known to date. It builds a monotone circuit with logarithmic depth and polynomial size that describe the access structure, and then this circuit is converted into a linear scheme. The upper bound is obtained by using the fact that every weighted threshold access structure admits an equivalent description with weights that are at most exponential [Mur71]. This result reveals that weighted threshold access structures are in a *privileged* position from the efficiency point of view, because most of the general access structures require linear schemes of share size $2^{n/3+o(n)}$ [BF20].

In recent works, Benhamouda, Halevi, and Stambler [BHS23] and Garg et al. [GJM$^+$23] explored a relaxed model, considering *ramp* weighted threshold access structures. In these access structures, there are privacy and correctness thresholds, and each party has a single weight. This setting admits schemes that are not perfect, allowing a reduction of the share size. The schemes in [BHS23] have a privacy threshold αW and a reconstruction threshold βW for some $0 < \alpha < \beta < 1$. In this setup, their first proposal is based on a rounding technique of weights that leads to a share size of $\frac{n}{\beta-\alpha}$. This technique can lead to smaller weights, but the error in the approximation cannot be bounded. For the second construction, they establish an interesting connection between wiretap channels. Choosing specific parameters, it is possible to get a scheme with $2^{-\lambda}$-statistical security and total share size $n \cdot \max\left\{\lambda^2, \mathrm{poly}(1/(\alpha-\beta))\right\}$.

The construction of Garg et al. [GJM$^+$23] uses as a primitive a scheme whose security is guaranteed by the Chinese Reminder Theorem [Mig83]. The privacy is statistical, and the gap between privacy and reconstructions thresholds depend on the security parameter λ. With this security relaxation, it is possible to improve the share size bound from Shamir's scheme, obtaining $O(W)$. The resulting scheme is not linear, but there are still ways to use it as a building block [GJM$^+$23].

We noticed that it is hard to give general bounds on the error of the approximation of weighted threshold access structures by (t, r)-ramp ones. For instance, if weights are close to W/n and the threshold is close to $W/2$, even a small gap in the thresholds may produce a significant error. Namely, in this case, (aW, bW)-ramp approximations for constants a, b may lead to an error tending to 1, which implies that the condition of being authorized or not changes for almost all sub-

sets. Despite that, in some scenarios, approximating by ramp weighted threshold access structures can be a good strategy. Compared to those previous works, our proposal realizes a perfect access structure and avoids moving to the ramp setting. This allows us to fully control the approximation error, knowing the number of inputs where the approximated access structure differs from the original one.

1.4 Organization

In Section 2 we lay out the preliminaries on the analysis of Boolean functions and secret sharing schemes. In Section 3 we present the technique for approximating monotone LTFs, obtaining Theorem 1.2 and proving the optimality of its bounds, i.e. Theorem 1.3. In Section 4 we construct the information-theoretic secret sharing scheme of Theorem 1.1, while in Section 5 we construct the computational secret sharing scheme of Theorem 1.4. The Appendix contains deferred proofs and definitions.

2 Preliminaries

Notation. We notate $\mathbb{N}$ and $\mathbb{R}_+$ for the sets of the non-negative integer and real numbers, respectively. For $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ as $[n]$. For a set S, we denote its cardinal as $|S|$. We denote vectors $\boldsymbol{x}$ using bold symbols, their i-th coordinates as x_i, their Euclidean norm as $\|\boldsymbol{x}\| = \sqrt{x_1^2 + \ldots + x_n^2}$, and their support as $\mathrm{supp}(\boldsymbol{x}) = \{i \in [n] \ : \ x_i \neq 0\}$. The unary vector is denoted by $\mathbf{1}^n \in \mathbb{R}^n$ and the zero vector is denoted by $\mathbf{0}^n$. For any vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we denote its scalar product as $\boldsymbol{x} \cdot \boldsymbol{y}$, its Hamming distance as $\mathrm{dist}_{\mathrm{Ham}}(\boldsymbol{x}, \boldsymbol{y}) = |\{i \in [n] \ : \ x_i \neq y_i\}|$, and we say that $\boldsymbol{x} \leq \boldsymbol{y}$ if and only if $x_i \leq y_i$ for all $i \in [n]$. For $x \in \{0,1\}$, $\overline{x}$ denotes its complementary, and for $\boldsymbol{x} \in \{0,1\}^n$, we set $\boldsymbol{x}^{\oplus i} = (x_1, \ldots, x_{i-1}, \overline{x_i}, x_{i+1}, \ldots, x_n)$.

Next, we define three functions over the reals. We set $\mathrm{sign} : \mathbb{R} \to \{-1, 1\}$ as the function with $\mathrm{sign}(x) = 1$ if and only if $x \geq 0$; we set $\mathrm{sign}_0 : \mathbb{R} \to \{0, 1\}$ as the function with $\mathrm{sign}_0(x) = 1$ if and only if $x \geq 0$; and $P_1 : \mathbb{R} \to [-1, 1]$ as the function with $P_1(x) = x$ if $x \in [-1, 1]$ and $P_1(x) = \mathrm{sign}(x)$ otherwise. For $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$ we define

- $\mathrm{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x}) = \mathrm{sign}_0(\boldsymbol{w} \cdot \boldsymbol{x} - \sigma)$,
- $\mathrm{LTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} - \sigma)$, and
- $\mathrm{LBF}(\boldsymbol{w}, \sigma)(\boldsymbol{x}) = P_1(\boldsymbol{w} \cdot \boldsymbol{x} - \sigma)$.

Throughout this work, all probability distributions $\mathbf{P}$ assume picking elements uniformly at random.

2.1 Analysis of Boolean Functions

In this section, we introduce all the definitions and results about Boolean functions needed for our purposes. First, we focus on the notions of weighted and linear threshold functions and their properties. Later, we state some results

about the Chow parameters. Finally, we introduce the notion of distance between Boolean functions and relate it with the Chow parameters. The statements and proofs of the switching lemmas, and the proof of Lemma 2.13 are deferred to Appendix A.

Weighted and Linear Threshold Functions. We start by presenting the main building block of our construction: weighted threshold functions.

Definition 2.1 (Weighted Threshold Function). *Let $\boldsymbol{w} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$. A Weighted Threshold Function (WTF) is a Boolean function $f : \{0,1\}^n \to \{0,1\}$ of the form $f(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$. The vector $(\boldsymbol{w}, \sigma)$ is said to represent f, while the vector $\boldsymbol{x} \in \{0,1\}^n$ is said to be* authorized *(resp.* forbidden*) if and only if $f(\boldsymbol{x}) = 1$ (resp. $f(\boldsymbol{x}) = 0$).*

Among all WTFs, we are interested in those that are monotone because they are in one-to-one correspondence with weighted threshold access structures. In this regard, the following remark gives a characterization of monotone WTFs.

Remark 2.2. A WTF given by $f(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$ is monotone if and only if there exists a representation of f with $w_i \geq 0$ for any $i \in [n]$. Indeed, if f is monotone increasing and $w_i < 0$, then f does not depend on x_i, and we can set its weight to 0.

An important result about monotone WTF is that they can be computed by polynomial size logarithmic depth monotone circuits of unbounded fan-in. This is the statement that follows.

Theorem 2.3 ([BW06]). *Every weighted threshold function is in $m\mathcal{AC}^1$.*

It is also usual to define Boolean functions by taking $\{-1,1\}^n$ as domain instead of $\{0,1\}^n$. Indeed, one can pass from one domain to the other by considering the bijection $\phi(x) = (-1)^x$ for $x \in \{0,1\}$ and extending it naturally to $\{0,1\}^n$. For this reason, we now define linear threshold functions, which are an analogue of WTF in the $\{-1,1\}^n$ domain.

Definition 2.4 (Linear Threshold Function). *Let $\boldsymbol{w} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$. A Linear Threshold Function (LTF) is a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ of the form $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$. The vector $(\boldsymbol{w}, \sigma)$ is said to represent f, while the vector $\boldsymbol{x} \in \{-1,1\}^n$ is said to be* authorized *(resp.* forbidden*) if and only if $f(\boldsymbol{x}) = 1$ (resp. $f(\boldsymbol{x}) = -1$).*

Related to this, using ϕ we can switch from any monotone WTF to an equivalent monotone LTF and vice versa without modifying the weight of any coordinate. More in detail, the switching lemmas deferred to Appendix A.1 imply that the weight vector representing a monotone LTF (resp. WTF) is not affected when converting it to a monotone WTF (resp. LTF). For this reason, along this work we will switch between both notions depending on which one is more useful for us at any given time.

We also need to introduce the following straightforward generalization of LTF.

Definition 2.5 (Linear Bounded Function). *Let $\boldsymbol{w} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$. A Linear Bounded Function (LBF) is a function $f : \{-1,1\}^n \to [1,1]$ of the form $f(\boldsymbol{x}) = \mathrm{LBF}(\boldsymbol{w},\sigma)(\boldsymbol{x})$. The vector $(\boldsymbol{w},\sigma)$ is said to represent f.*

Next, we define an important notion in the context of monotone LTFs: the influence of each coordinate.

Definition 2.6. *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a Boolean function. For any $i \in [n]$, the influence of the i-th coordinate on f is the fraction of input values in which it affects the output, i.e. $\mathbf{Inf}_i[f] = \mathbf{P}[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})]$.*

We end this section by stating two useful results about the weights of monotone LTFs. The first one shows that any monotone LTF can be expressed by a monotone formula of size polynomial in the sum of the weights. The second is a known upper bound on the size of the weights of any monotone LTF.

Theorem 2.7 ([Ser04]). *For any $\boldsymbol{w} \in \mathbb{R}^n_+$ and $\sigma \in \mathbb{R}$ the monotone LTF given by $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w},\sigma)(\boldsymbol{x})$ has a monotone formula of size $O(W^{5.3})$, where $W = \boldsymbol{w} \cdot \mathbf{1}^n$.*

Theorem 2.8 ([Mur71]). *For any monotone LTF f with n variables there exist $w_1,\ldots,w_n,\sigma \in \mathbb{N}$ smaller than $2^{\lceil n\log(n)\rceil}$ such that $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w},\sigma)(\boldsymbol{x})$, where $\boldsymbol{w} = (w_1,\ldots,w_n)$.*

Chow Parameters. We first give the definition of the Chow parameters. Let $\mathbf{E}$ denote the expectancy of a discrete random variable.

Definition 2.9 (Chow Parameters). *The Chow parameters of a function $f : \{-1,1\}^n \to \mathbb{R}$ are the $n+1$ values $\hat{f}(0) = \mathbf{E}[f(\boldsymbol{x})]$ and $\hat{f}(i) = \mathbf{E}[f(\boldsymbol{x})x_i]$ for any $i \in [n]$, taking uniform distribution on its domain. The Chow vector of f is $\boldsymbol{\chi}_f = (\hat{f}(0),\ldots,\hat{f}(n))$.*

Chow parameters are a particular case of a much more general family known as the Fourier coefficients. More in detail, each function $f : \{-1,1\}^n \to \mathbb{R}$ can be uniquely expressed as a multilinear polynomial whose coefficients are defined as the Fourier coefficients of f. In this setting, it can be shown that the Chow parameters simply correspond to the Fourier coefficients of degree 0 and 1. The book of O'Donnell [O'D14] offers a detailed discussion of this topic.

Following the later construction, we could also have defined the Fourier coefficients (and in particular the Chow parameters) for Boolean functions in the $\{0,1\}$ domain. However, in this case we can no longer define the Chow parameters in terms of expectations as is done in Definition 2.9. This is because the basis in which the Fourier coefficients are constructed is not orthonormal. Therefore, when it comes to Chow parameters, we always consider Boolean functions in the $\{-1,1\}$ domain.

In this regard, recall that for monotone LTFs we can always move from one domain to the other thanks to Lemma A.1 and Lemma A.2. Hence, by an abuse

of notation, when we consider the Chow parameters of a WTF, we refer to the Chow parameters of its LTF analogue.

An important result regarding the Fourier coefficients is the following one, known as Plancherel's Theorem.

Theorem 2.10 (Plancherel's Theorem [O'D14]). *For any functions* $f, g : \{-1,1\}^n \to \mathbb{R}$ *we have that* $\mathbf{E}[f(\boldsymbol{x})g(\boldsymbol{x})] = \sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S)$.

Nowadays, Chow parameters (and by extension the Fourier coefficients) have become one of the most used tools for the study of Boolean functions. In the case of LTFs, this is mainly motivated by the next theorem, known as Chow's Theorem.

Theorem 2.11 (Chow's Theorem [Cho61]). *Any LTF is uniquely determined within the space of Boolean functions by its Chow parameters.*

Moreover, in the case of monotone Boolean functions, the Chow parameters have an additional interpretation as gauges of the influence of each coordinate. This is stated in the following proposition.

Proposition 2.12 ([O'D14]). *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be a monotone Boolean function. Then* $\hat{f}(i) = \mathbf{Inf}_i[f]$ *for any* $i \in [n]$.

We conclude this section by presenting a useful lemma relating monotonicity, projections and Chow parameters. Its proof is deferred to Appendix A.2.

Lemma 2.13. *Let* $f : \{-1,1\}^n \to \mathbb{R}$ *be a monotone function and* $g(\boldsymbol{x}) = P_1(f(\boldsymbol{x}))$. *Then* $\hat{f}(i) \geq \hat{g}(i) \geq 0$ *for any* $i \in [n]$.

To simplify the notation and without loss of generality, from now on we will assume that any LTF has the weights sorted in decreasing order, which immediately implies a decreasing order in its Chow parameters except for $\hat{f}(0)$.

Distance. The concept of distance between Boolean functions gives a way to measure the similarity of two access structures. Hence, we introduce it to formally define the notion of closeness between monotone LTFs.

Definition 2.14 (Function Distance). *Let* X *be a finite set. The distance between two functions* $f, g : X \to \mathbb{R}$ *is defined as* $\mathrm{dist}(f,g) = \mathbf{E}[|f(\boldsymbol{x}) - g(\boldsymbol{x})|]$. *If* $\mathrm{dist}(f,g) < \epsilon$ *we say that* f *and* g *are* ϵ*-close. Moreover, the Chow distance between* f *and* g *is defined as* $\mathrm{dist}_{\mathrm{Chow}}(f,g) = \|\boldsymbol{\chi}_f - \boldsymbol{\chi}_g\|$.

Remark 2.15. Notice that if f, g are WTFs then $\mathrm{dist}(f,g) = \mathbf{P}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$, while if f, g are LTFs then $\mathrm{dist}(f,g) = 2\mathbf{P}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$.

Both notions of distance are closely related. In particular, for the case of LTFs each of them can be bounded in terms of the other. This is stated in the theorems that follow.

Theorem 2.16 ([OS11]). *For every $f, g : \{-1,1\}^n \to \mathbb{R}$, it holds that* $\mathrm{dist}_{\mathrm{Chow}}(f,g) \leq 2\sqrt{\mathrm{dist}(f,g)}$.

Theorem 2.17 ([DDFS14]). *Let f be a LTF and let $g : \{-1,1\}^n \to [-1,1]$ be any function. If* $\mathrm{dist}_{\mathrm{Chow}}(f,g) \leq \epsilon$, *then* $\mathrm{dist}(f,g) \leq 2^{-\Omega(\sqrt[3]{\log \frac{1}{\epsilon}})}$.

Theorem 2.16 and Theorem 2.17 establish a relation between the upper bounds given by the distances of LTFs and their Chow distances, which provides a strategy to check closeness between LTFs simply by looking at their Chow parameters. Indeed, this is the key observation that we will exploit in the next section to approximate monotone LTFs.

2.2 Secret Sharing Schemes

For convenience, we work with access structures described by monotone Boolean functions, which is equivalent to work with monotone increasing families of subsets.

Definition 2.18 (Access Structure). *An n-party access structure is a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $f(\mathbf{0}^n) = 0$ and $f(\mathbf{1}^n) = 1$.*

If $f(\boldsymbol{x}) = 1$, we say that the set $A = \mathrm{supp}(\boldsymbol{x})$ is authorized, *and else we say that A is* forbidden.

Definition 2.18 implies that non-constant monotone WTF are a particular case of access structures. In this context, they are also called *weighted threshold access structures*.

Remark 2.19. In the case of a weighted threshold access structure given by $f(\boldsymbol{x}) = WTF(\boldsymbol{w}, \sigma)(\boldsymbol{x})$, the family of authorized subsets corresponds to

$$\Gamma = \{A \subseteq [n] \ : \ \sum_{i \in A} w_i \geq \sigma\}.$$

Moreover, note that if two functions f and f' are ϵ-close, the corresponding monotone families of subsets Γ and Γ' satisfy $|\Gamma \cup \Gamma'| - |\Gamma \cap \Gamma'| < \epsilon 2^n$.

Secret sharing schemes admit information-theoretic and computational constructions. We first define information-theoretic secret sharing schemes.

Definition 2.20 (Information-Theoretic Secret Sharing Scheme). *Let $\mathcal{S}$ be a finite set and let f be an access structure over n parties. A secret sharing scheme for f is a pair of a randomized algorithm* Share *and deterministic algorithms* $\mathsf{Reconstruct}_A$ *such that*

- ***Perfect correctness.*** *For any secret $s \in \mathcal{S}$ and any authorized set A it holds that*

$$\mathbf{P}[s = \mathsf{Reconstruct}_A(\mathsf{Share}(s)_A)] = 1,$$

where $\mathsf{Share}(s)_A$ denotes the restriction of the output of $\mathsf{Share}(s)$ to the parties in A.

– ***Perfect privacy.*** *For any secrets $s, s' \in \mathcal{S}$, any forbidden set B, and any possible set of shares $\{s_i\}_{i\in B}$ it holds that*

$$\mathbf{P}[\{s_i\}_{i\in B} = \mathsf{Share}(s)_B] = \mathbf{P}[\{s_i\}_{i\in B} = \mathsf{Share}(s')_B].$$

As of today, the best information-theoretic secret sharing scheme for general weighted threshold access structures in terms of the share size is the following result. It is due to the work of Beimel and Weinreb [BW06] and consists in describing monotone circuits that compute any threshold function and later applying monotone-circuits-to-secret-sharing compilers [BL88].

Theorem 2.21 ([BW06]). *Every weighted threshold access structure over n parties has an information-theoretic secret sharing scheme with with share size $n^{O(\log(n))}$.*

We now state the definition of computational secret sharing schemes. The security of computational secret sharing schemes is given in terms of a game between an adversary and a challenger. For the sake of completeness, we defer its definition to Appendix B.1.

Definition 2.22 (Computational Secret Sharing Scheme). *Let $\mathcal{S}$ be a finite set, let $\lambda \in \mathbb{N}$ be the security parameter, and let f be an access structure over n parties. A computational secret sharing scheme for f is a pair of a randomized polynomial-time algorithm* Share *and deterministic polynomial-time algorithms* $\mathsf{Reconstruct}_A$ *such that*

– ***Correctness.*** *For any secret $s \in \mathcal{S}$ and any authorized set A it holds that*

$$\mathbf{P}[s = \mathsf{Reconstruct}_A(\mathsf{Share}(s, \mathbf{1}^\lambda)_A)] = 1,$$

where $\mathsf{Share}(s, \mathbf{1}^\lambda)_A$ denotes the restriction of the output of $\mathsf{Share}(s, \mathbf{1}^\lambda)$ to the parties in A.

– ***Privacy.*** *The scheme is secure if any polynomial-time adversary succeeds in breaking the scheme with negligible probability.*

3 Approximation of Monotone Linear Threshold Functions

The aim of this section is to approximate a monotone LTF with another monotone LTF with smaller integer weights. To do so, we adapt to the monotone setup the work of De et al. [DDFS14], in which an algorithm for approximating LTFs is presented. To simplify the presentation of this and the following results, we use $\tilde{O}$ notation, which ignores polylogarithmic factors. The main result of this section is the following theorem.

Theorem 3.1 (Theorem 1.2 restated). *Let $\kappa(\epsilon) = 2^{-O(\log^3(\frac{1}{\epsilon}))}$ and let $\mu(n) \in \mathbb{R}$ be a function that satisfies $\mu(n) \geq 2\sqrt{n+1}$. For $\delta, \epsilon \in (0,1)$, there exists a randomized algorithm* $\mathsf{ApproximateLTF}$ *that given a monotone LTF over n variables f with $\kappa(\epsilon) \leq \hat{f}(n)\mu(n)$, outputs with probability $1-\delta$ a monotone function $g(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{v}, v_0)(\boldsymbol{x})$ with the following properties:*

1. *g is ϵ-close to f,*
2. *$\boldsymbol{v} \in \mathbb{N}^n$, $v_0 \in \mathbb{Z}$, and $\|\boldsymbol{v}\| = O\left(\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$.*

Further, the algorithm runs in time $\tilde{O}\left(n\mu(n)^2 \text{poly}\left(\frac{1}{\kappa(\epsilon)}\right)\log\left(\frac{1}{\delta}\right)\right)$.

To prove Theorem 3.1 it suffices to construct a candidate algorithm for ApproximateLTF and check that it satisfies all the requirements from the statement. We do this in two steps. First, we assume the existence of a similar algorithm ApproximateLBF whose output is a monotone LBF and show how to use it to construct the desired ApproximateLTF algorithm. Second, we prove that this ApproximateLBF algorithm exists.

In particular, the proof of Theorem 3.1 relies on the following result.

Theorem 3.2. *Let $\mu(n) \in \mathbb{R}$ be a function that satisfies $\mu(n) \geq 2\sqrt{n+1}$. For $\delta, \epsilon \in (0,1)$, there exists a randomized algorithm* ApproximateLBF *that given a monotone LTF over n variables f with $\epsilon \leq \hat{f}(n)\mu(n)$, outputs with probability $1-\delta$ a monotone function $g(\boldsymbol{x}) = \text{LBF}(k\boldsymbol{v}, kv_0)(\boldsymbol{x})$ with the following properties:*

1. *$\text{dist}_{\text{Chow}}(f,g) \leq 3\epsilon$,*
2. *$k \in \mathbb{R}, \boldsymbol{v} \in \mathbb{N}^n$, $v_0 \in \mathbb{Z}$, and $\|\boldsymbol{v}\| = O\left(\frac{\mu(n)}{\epsilon^3}\right)$.*

Further, the algorithm runs in time $\tilde{O}\left(n\frac{\mu(n)^2}{\epsilon^4}\log\left(\frac{1}{\delta}\right)\right)$.

We start by proving Theorem 3.1 using Theorem 3.2. The proof of Theorem 3.2 is more involved, and it deferred to Section 3.1.

Proof of Theorem 3.1. A direct application of Theorem 3.2 guarantees that in a running time of $\tilde{O}\left(n\mu(n)^2\text{poly}\left(\frac{1}{\kappa(\epsilon)}\right)\log\left(\frac{1}{\delta}\right)\right)$ we obtain a monotone LBF $g(\boldsymbol{x}) = \text{LBF}(\boldsymbol{v}, v_0)$ such that $\text{dist}_{\text{Chow}}(f,g) \leq 3\kappa(\epsilon)$ with probability $1-\delta$. From there, adjusting properly the constants of $\kappa(\epsilon)$ and applying Theorem 2.17 we get that $\text{dist}(f,g) \leq \frac{\epsilon}{2}$.

Now, defining $f'(\boldsymbol{x})$ as $f'(\boldsymbol{x}) = \text{LTF}(\boldsymbol{v}, v_0)(\boldsymbol{x})$ it is straightforward to check that f' is monotone and $\text{dist}(f,f') \leq 2\text{dist}(f,g) \leq \epsilon$. Moreover, again by Theorem 3.2 we have that $\|\boldsymbol{v}\| = O\left(\frac{\mu(n)}{\kappa(\epsilon^3)}\right) = O\left(\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$. □

At this point, to prove Theorem 3.1 it only remains to prove Theorem 3.2.

3.1 Proof of Theorem 3.2

We first present a high-level overview of the construction of the ApproximateLBF algorithm. As in the work from De et al. [DDFS14], it mainly relies in the straightforward observation that the function $\hat{f}(0) + \sum_{i=1}^n \hat{f}(i)x_i$ has exactly the same Chow parameters as the input LTF f.

Starting from there, the function $g(\boldsymbol{x}) = P_1(\hat{f}(0) + \sum_{i=1}^{n} \hat{f}(i)x_i)$ seems a reasonable candidate for being the monotone LBF output by the algorithm because it is build based on the Chow parameters of f. However, taking the projection of a function leads to a modification of the original Chow parameters, so our candidate g may not satisfy the desired condition on the Chow distance.

To solve this problem, we could try to correct this gap in the Chow distance following a similar procedure as before. In particular, we can construct a function h whose Chow parameters correspond to the difference between $\boldsymbol{\chi}_f$ and $\boldsymbol{\chi}_g$, i.e. $h(\boldsymbol{x}) = \hat{f}(0) - \hat{g}(0) + \sum_{i=1}^{n}(\hat{f}(i) - \hat{g}(i))x_i$, and add it to g with the aim of obtaining a better output candidate $P_1(g + h)$.

At this point, we face again the problem of checking up to which point the projection operation has modified the Chow parameters of our output candidate. Hence, we can repeat the previous procedure iteratively seeking that at each step we will get closer to the desired result.

In general terms, ApproximateLBF algorithm implements the idea we have just presented. Nevertheless, some minor changes are introduced to deal with technicalities regarding monotonicity and the bounds on the weights and the running time. The ApproximateLBF algorithm is depicted in detail in Algorithm 1, while Algorithm 2 presents an auxiliary method.

Notice that in the work of De et al. [DDFS14] the context is slightly different. They propose this algorithm to solve the approximate Chow problem, i.e. given the approximate Chow parameters of an unknown LTF they try to recover it by constructing an ϵ-close LTF to it. The two main differences with our setup are that we already know in advance the original monotone LTF and that in our case the constructed LTF has to be monotone. Hence, all the changes made in the statement and proof of Theorem 3.2 are due to this differences.

Algorithm 1 ApproximateLBF

Input: Monotone $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}, w_0)(\boldsymbol{x})$, Chow parameters $\boldsymbol{\chi}_f$, and $\delta, \epsilon \in (0, 1)$

Output: Monotone $g(\boldsymbol{x}) = \mathrm{LBF}(k\boldsymbol{v}, kv_0)(\boldsymbol{x})$ with $\mathrm{dist}_{\mathrm{Chow}}(f, g) \leq 3\epsilon$, $\|\boldsymbol{v}\| = O\left(\frac{\sqrt{n}}{\epsilon^3}\right)$

Require: $\mu(n) \geq 2\sqrt{n+1}$ and $\epsilon \leq \hat{f}(n)\mu(n)$

$g' \leftarrow 0$, $g \leftarrow P_1(g')$
$\boldsymbol{\chi}_{\tilde{g}} \leftarrow \mathsf{VectorApprox}(\boldsymbol{\chi}_f, \boldsymbol{\chi}_g, \epsilon)$
while $\|\boldsymbol{\chi}_f - \boldsymbol{\chi}_{\tilde{g}}\| \leq 2\epsilon$ **do**
    $h \leftarrow \sum_{i=0}^{n}(\hat{f}(i) - \tilde{g}(i))x_i$
    $g' \leftarrow g' + \frac{h}{2}$
    $g \leftarrow P_1(g')$
    Compute $\boldsymbol{\chi}_g$ with precision $\frac{\epsilon}{2\mu(n)}$ ▷ Chow parameters of g
    $\boldsymbol{\chi}_{\tilde{g}} \leftarrow \mathsf{VectorApprox}(\boldsymbol{\chi}_f, \boldsymbol{\chi}_g, \epsilon)$ ▷ Rounding to ensure integer difference
end while
return g

To prove Theorem 3.2 it suffices to check that ApproximateLBF (Algorithm 1) satisfies all the conditions of the statement. In comparison to the work of De

Algorithm 2 VectorApprox

Input: Chow parameters $\boldsymbol{\chi}_f, \boldsymbol{\chi}_g \in \mathbb{R}^{n+1}$ and $\epsilon \in (0,1)$
Output: Vector $\boldsymbol{\chi}_{\tilde{g}} \in \mathbb{R}^{n+1}$

1: **for** $i = 0, \ldots, n$ **do**
2: $\quad \tilde{g}(i) \leftarrow$ the closest value to $\hat{g}(i)$ such that $\hat{f}(i) - \tilde{g}(i) = k\frac{\epsilon}{\mu(n)}$ with $k \in \mathbb{Z}$
3: **end for**
4: **return** $\boldsymbol{\chi}_{\tilde{g}} = (\tilde{g}(0), \ldots, \tilde{g}(n))$

et al. [DDFS14], apart from slightly differences in the bounds of some norms, our proof requires a more fine-grained analysis to ensure that the resulting approximate function is also monotone. For this reason, we divide it into several propositions: one for checking the monotonicity of the output, another to ensure the halting of the algorithm, and a last one for the bounds on the weights and the running time.

We start by proving that the output of ApproximateLBF (Algorithm 1) corresponds to a monotone LBF satisfying the gap in the Chow distance. This is stated in Proposition 3.3 and its proof is deferred to Appendix C.1. Indeed, this is the part in which our work differs the most from the one of De et al. [DDFS14], since it is where we impose the monotonicity condition on the output.

Proposition 3.3. ApproximateLBF *(Algorithm 1) outputs with probability* $1-\delta$ *a monotone LBF* g *such that* $\mathrm{dist}_{\mathrm{Chow}}(f,g) \leq 3\epsilon$.

Proposition 3.3 ensures that the output of ApproximateLBF (Algorithm 1) is a monotone LBF satisfying the required distance in the Chow parameters. However this is a meaningless result unless we prove that ApproximateLBF (Algorithm 1) always halts, since apparently nothing prevents the algorithm to stay indefinitely in the main loop (steps 4-8). The next proposition shows that this situation can not happen. Its proof is almost analogous to the one from De et al. [DDFS14] and we defer it to Appendix C.2.

Proposition 3.4. *The main loop of* ApproximateLBF *(Algorithm 1, steps 4-8) requires at most* $\frac{1}{\epsilon^2}$ *iterations.*

Finally, the remaining proposition gives the bound on the weights and the running time. It is a crucial result, since the bound on the weights is what later enables the construction of secret sharing schemes for weighted threshold access structures with small share size. Moreover, the bound on the running time shows the feasibility of this strategy.

Proposition 3.5. *The monotone function* $g(\boldsymbol{x}) = \mathrm{LBF}(k\boldsymbol{v}, kv_0)(\boldsymbol{x})$ *output by* ApproximateLBF *(Algorithm 1) has the following properties:*

1. $k \in \mathbb{R}, \boldsymbol{v} \in \mathbb{N}^n$, $v_0 \in \mathbb{Z}$, *and* $\|\boldsymbol{v}\| = O\left(\frac{\mu(n)}{\epsilon^3}\right)$.
2. *Its running time is* $\tilde{O}\left(n\frac{\mu(n)^2}{\epsilon^4}\log\left(\frac{1}{\delta}\right)\right)$.

Again, the full proof of Proposition 3.5 is included in Appendix C.3. At this point, to prove Theorem 3.2 it suffices to combine the results of Proposition 3.3, Proposition 3.4 and Proposition 3.5.

3.2 Remarks on the Error and the Weight Bound of Theorem 3.1

Theorem 3.1 sets a method for approximating monotone LTFs up to any accuracy and gives a bound on the resulting weights in terms of the number of variables and the error. However, there are some additional properties that any approximation technique must satisfy to be suitable for usage, such as preserving the order of the weights or having optimal bounds.

In this section, we show that our proposal fulfills all these desired requirements. First, we show that the error has a negligible impact on the original hierarchy of the coordinates. Later, we prove the optimality of the algorithm with respect to the weight bound. Finally, we perform an analysis on the trade-off between both notions: the error and the size of the weights.

On the Error and the Preservation of the Weights Hierarchy. One of the biggest concerns when approximating any monotone LTF is the error it produces and how it can affect the hierarchy of the weights. In this regard, we must ensure that the error produced does not cause a drastic change in the impact of each coordinate to the output of the function. In other words, any approximating procedure must maintain the influence of the coordinates unchanged. The next result shows that our technique technique satisfies this property.

Theorem 3.6. *Let f, g be two $o(1)$-close monotone LTFs. Then, $|\mathbf{Inf}_i[f] - \mathbf{Inf}_i[g]| = o(1)$ for any $i \in [n]$.*

Proof. By Proposition 2.12 it is enough to prove that $\mathrm{dist}_{\mathrm{Chow}}(f, g) = o(1)$. Now, Theorem 2.16 states that $\mathrm{dist}_{\mathrm{Chow}}(f, g) \leq 2\sqrt{\mathrm{dist}(f, g)}$, so using that $\mathrm{dist}(f, g) = o(1)$ we are done. □

Furthermore, we can go even further and ask not only that the influence of each coordinate is preserved but also that the hierarchy of the weights is maintained. That is, we demand that whenever one coordinate has a greater weight than another in the original function, the same happens after the approximation. This can be achieved simply by rearranging the weights of the approximation function at the end of the process. In this regard, the next lemma proves that this additional modification does not increase the error of the approximation.

Lemma 3.7. *Let f be any monotone LTF, let $g(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}, w_0)(\boldsymbol{x})$ be any monotone LTF ϵ-close to f, and let $h(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}', w_0)(\boldsymbol{x})$, where $\boldsymbol{w}'$ is the weight vector $\boldsymbol{w}$ sorted in decreasing order. It holds that $\mathrm{dist}(f, h) \leq \epsilon$.*

Proof. First, notice that it suffices to prove the statement for the case where the vector $\boldsymbol{w}$ is already sorted except for any two coordinates $i < j$, since any other case can be reduced to this one.

Now, to show that $\mathrm{dist}(f,h) \leq \epsilon$ we must show that for any $\boldsymbol{x} \in \{-1,1\}^n$ with $x_i, x_j = -1$ it holds that

$$\left|\left\{f(\boldsymbol{y}) = g(\boldsymbol{y}) \ : \ \boldsymbol{y} \in \{\boldsymbol{x}^{\oplus i}, \boldsymbol{x}^{\oplus j}\}\right\}\right| \leq \left|\left\{f(\boldsymbol{y}) = h(\boldsymbol{y}) \ : \ \boldsymbol{y} \in \{\boldsymbol{x}^{\oplus i}, \boldsymbol{x}^{\oplus j}\}\right\}\right|.$$

Since both f and g, are monotone Boolean functions, each of them has only three possible outputs for any pair of inputs $\boldsymbol{x}^{\oplus i}, \boldsymbol{x}^{\oplus j}$. More in detail, either $f(\boldsymbol{x}^{\oplus i}) = f(\boldsymbol{x}^{\oplus j}) = -1$, $f(\boldsymbol{x}^{\oplus i}) = f(\boldsymbol{x}^{\oplus j}) = 1$, or $f(\boldsymbol{x}^{\oplus i}) = 1 > f(\boldsymbol{x}^{\oplus j}) = -1$ (and similarly for g). Therefore, we are left with a total of 9 different cases.

We now prove the case where

$$f(\boldsymbol{x}^{\oplus i}) = 1,\ f(\boldsymbol{x}^{\oplus j}) = -1,\ g(\boldsymbol{x}^{\oplus i}) = -1, \text{ and } g(\boldsymbol{x}^{\oplus j}) = 1.$$

By hypothesis, we have that

$$\left|\left\{f(\boldsymbol{y}) = g(\boldsymbol{y}) \ : \ \boldsymbol{y} \in \{\boldsymbol{x}^{\oplus i}, \boldsymbol{x}^{\oplus j}\}\right\}\right| = 0.$$

Moreover, by the definition of h we have that $g(\boldsymbol{x}^{\oplus i}) = -1$ implies that $h(\boldsymbol{x}^{\oplus j}) = -1$, and that $g(\boldsymbol{x}^{\oplus j}) = 1$ implies that $h(\boldsymbol{x}^{\oplus i}) = 1$. Therefore, we get that

$$\left|\left\{f(\boldsymbol{y}) = h(\boldsymbol{y}) \ : \ \boldsymbol{y} \in \{\boldsymbol{x}^{\oplus i}, \boldsymbol{x}^{\oplus j}\}\right\}\right| = 2,$$

which proves this case.

The remaining eight cases follow a similar procedure. □

Therefore, we conclude that our technique not only approximates any monotone LTFs up to a given accuracy, but it also preserves the original hierarchy on the weights and the influence of each coordinate.

Optimality of the Weight Bound. One may wonder if the weight bound of Theorem 3.1 can be lowered in terms of its dependency on n or $\frac{1}{\epsilon}$. We answer this question by showing that the bound is optimal in terms of n and is close of being optimal in terms of $\frac{1}{\epsilon}$. In particular, we prove the following theorem.

Theorem 3.8 (Theorem 1.3 restated). *Let* $\kappa = \max\{\sqrt{n}, \left(\frac{1}{\epsilon}\right)^{\Omega\left(\log\left(\log\left(\frac{1}{\epsilon}\right)\right)\right)}\}$. *There does not exist an algorithm* OptimalApproximateLTF *such that for any monotone LTF over* n *variables* f *and* $\epsilon \in (0,1)$ *outputs a monotone function* $g(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{v}, v_0)(\boldsymbol{x})$ *with the following properties:*

1. g *is* ϵ*-close to* f,
2. $\boldsymbol{v} \in \mathbb{Z}^n$ *and* $\|\boldsymbol{v}\| = O(\kappa)$.

Remark 3.9. In the statement of Theorem 3.1 the weight bound, with respect to n, corresponds to any $\mu(n) \geq 2\sqrt{n+1}$. Therefore, when we look at the optimality of the bound in Theorem 3.8 we can replace $\mu(n)$ by $\sqrt{n}$.

As we did with Theorem 3.1, we split the proof of Theorem 3.8 in two steps. First, we prove that the bound is optimal in terms of its dependency on n. Later, we study the optimality of the bound with respect to ϵ.

To show that the bound can not be improved in terms of n it suffices to give an explicit monotone LTF f over n variables and a particular value of ϵ such that any LTF g ϵ-close to f has some weight of size $\Omega(\sqrt{n})$.[3] This is precisely what Servedio did in a previous work [Ser06], where he proved the result that follows.

Theorem 3.10 ([Ser06]). *Let $\boldsymbol{w} = (1, \dots, 1, n) \in \mathbb{Z}^n$, let f be the monotone LTF given by $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}, n)(\boldsymbol{x})$, and let $g : \{-1, 1\}^n \to \{-1, 1\}$ be an $\frac{1}{10}$-close LTF to f. Then any integer representation of g must have some weight of size $\Omega(\sqrt{n-1})$.*

Next, we move to study the optimality of the bound in terms of $\frac{1}{\epsilon}$. To obtain the bound stated in Theorem 3.8 we rely on a particular LTF introduced by Håstadt [Hås94] that requires integer weights of size $2^{\Omega(n \log(n))}$.

Theorem 3.11. *There exist a monotone LTF and $\epsilon \in (0, 1)$ such that any monotone LTF g ϵ-close to f has some weight of size $\left(\frac{1}{\epsilon}\right)^{\Omega\left(\log\log\left(\frac{1}{\epsilon}\right)\right)}$.*

Proof. We argue by contradiction. Suppose that for any monotone LTF f and $\epsilon \in (0, 1)$ we can construct a monotone LTF g ϵ-close to f with weights of size $\left(\frac{1}{\epsilon}\right)^{o\left(\log\log\left(\frac{1}{\epsilon}\right)\right)}$. Now, let f be the LTF introduced by Håstadt [Hås94] that requires integer weights of size $2^{\Omega(n \log(n))}$. Without loss of generality, we can suppose that f is also monotone.[4] Then, for any $\epsilon < \frac{1}{2^n}$ by hypothesis we can construct a monotone LTF g with integer weights of size $2^{o(n \log(n))}$ that is ϵ-close to f. Since there are only $2^n < \frac{1}{\epsilon}$ distinct input values, this implies that $f = g$. But this clearly contradicts the lower bound on the weights of the Håstadt function f. □

If we consider Theorem 3.1 as a strategy to reduce the weights of a given monotone LTF, we can interpret Theorem 3.8 not only as the limitations of this technique, but also as the limitations of any approximation technique based on the reduction of the weights, as for example rounding. However, since Theorem 3.1 is a general result, it is possible that some monotone LTFs admit low-weight approximators with smaller weights.

[3] Since we are looking for a lower bound in terms of n, note that we can set in advance the approximation error ϵ to a concrete value and the approximate LTF g does not need to be monotone.

[4] By definition, any LTF can be converted into a monotone LTF simply by flipping some of its input variables. If f is not monotone, it suffices to define f' as the monotone LTF obtained from f by this procedure. It is clear that the magnitude of the weights of f' is the same as that of the weights of f.

Trade-Off between the Error and the Weight Bound. We end this section by studying the trade-off between the size of the weights and the error of Theorem 3.1. Since the weight bound of Theorem 3.1 depends on $\frac{1}{\epsilon}$, the more accurate the approximation, the higher the weight bound will be. Moreover, this factor is of the form quasipoly $\left(\frac{1}{\epsilon}\right)$, which implies that the weight bound grows faster than the accuracy of the approximation does.

In this regard, note that, at the cost of increasing the weight bound, we can make the error ϵ as small as desired because there is no lower restriction apart from the trivial $0 < \epsilon$. However, we can not increase ϵ freely since Theorem 3.1 requires that $2^{-O(\log^3\left(\frac{1}{\epsilon}\right))} < \hat{f}(n)\mu(n)$. This can be annoying in case we want to minimize the size of the weights.

To overcome this limitation, we can try to increase ϵ by pushing up the value of its upper bound $\hat{f}(n)\mu(n)$. In that sense, the factor $\hat{f}(n)$ is given by the original LTF and Proposition 2.12 implies that, in the worst case, it can be equal to $\frac{1}{2^{n-1}}$. Hence, our only option is to increase the value of $\mu(n)$, which is lower bounded by $2\sqrt{n+1}$.

Nevertheless, note that $\mu(n)$ also appears as a linear factor of the weight bound. For this reason, we must be careful when increasing the value of $\mu(n)$ with the aim of decreasing the weight bound, since pushing it too high may go against our interests.

A more fine-grained analysis of this trade-off will appear later in Section 4 during the proof of Theorem 4.3. As we will see, a better result can be obtained by discarding the Chow parameters with the lowest values.

4 Secret Sharing Schemes for Approximate Weighted Threshold Access Structures

In this section we apply the results on low-weight approximators for monotone LTFs to construct information-theoretic secret sharing schemes for weighted threshold access structures with small share size. First, we introduce our proposal and discuss some alternatives. Later, we compare it with state-of-the-art solutions.

4.1 Scheme Construction

The main result of this section is the following theorem.

Theorem 4.1. *Let $\kappa(\epsilon) = 2^{-O(\log^3\left(\frac{1}{\epsilon}\right))}$ and let $\mu(n) \in \mathbb{R}$ be a function that satisfies $\mu(n) \geq 2\sqrt{n+1}$. For any weighted threshold access structure over n parties f and $\epsilon \in (0,1)$ with $\kappa(\epsilon) \leq \hat{f}(n)\mu(n)$, there exists a weighted threshold access structure over n parties ϵ-close to f admitting an information-theoretic secret sharing scheme with share size $\tilde{O}\left(\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$.*

Proof. First, recall that due to the switching lemmas of Appendix A.1 we can work indistinctly with f and its equivalent monotone WTF. Hence, in this proof we will not make any distinction between them.

Now, given $\mu(n) > 2\sqrt{n+1}$ and $\epsilon \in \left(0, \hat{f}(n)\mu(n)\right]$ we apply Theorem 3.1 to obtain a monotone function $g(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{v}, v_0)(\boldsymbol{x})$ that is ϵ-close to f. Moreover, it holds that $\boldsymbol{v} \in \mathbb{N}^n$, $v_0 \in \mathbb{Z}$, and $\|\boldsymbol{v}\| = O\left(\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$.

By construction, g is a weighted threshold access structure with weights $\boldsymbol{v}$, threshold v_0', and with $\mathrm{dist}(f, g) < \epsilon$. In addition, since we know that $\|\boldsymbol{v}\| = O\left(\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$ it suffices to use Shamir's virtualization technique to obtain the desired secret sharing scheme for g. □

Remark 4.2. Since we use Shamir's virtualization technique, the secret sharing scheme constructed in Theorem 4.1 is also linear. This implies that our proposal has homomorphic properties and efficient Share and Reconstruction algorithms.

Following the arguments of Section 3.2, it seems inevitable that to obtain a small upper bound on the share size in Theorem 3.1, the Chow parameters associated to the weighted threshold access structure must be big enough. Otherwise, the inequality $\kappa(\epsilon) \leq \hat{f}(n)\mu(n)$ would imply that either ϵ is too small or $\mu(n)$ is too big, which in both cases would lead to an increase in the upper bound on the share size.[5]

However, if some of the Chow parameters are small, we can still obtain a small upper bound on the share size simply by discarding those parties from the original weighted threshold access structure. More in detail, thanks to Proposition 2.12 we can view each Chow parameter as the influence of a specific party in the access structure. Therefore, if some of the Chow parameters are small, we can guarantee that removing them from the access structure would not modify it too much. In this way, we can always control the value of the $\hat{f}(n)$ that appears in the statement of Theorem 3.1.

As a consequence of combining this observation with Theorem 4.1 we obtain the following result.

Theorem 4.3 (Theorem 1.1 restated). *Let $k \in \mathbb{N}$. For any weighted threshold access structure over n parties f there exists a weighted threshold access structure over n parties $\frac{1}{\log^k(n)}$-close to f admitting an information-theoretic secret sharing scheme with share size $n^{1+o(1)}$.*

Proof. Let $\boldsymbol{w}$ be the weight vector of f, σ its threshold value, and let $l \in [n]$ be the maximum value such that $\hat{f}(l) \geq \frac{1}{2n\log^k(n)}$.

[5] The fact that the upper bound on the share size given by Theorem 3.1 is high does not rule out the possibility that the shares obtained via the approximation technique are small. In other words, it is possible to obtain a scheme with short shares by using ApproximateLTF algorithm even if $\hat{f}(n)$ is small.

We consider the weighted threshold access structure over l parties f' given by the weight vector $(w_1, \ldots, w_l)$ and threshold σ. Proposition 2.12 implies that $\mathrm{dist}(f, f') \leq \frac{n}{2n\log^k(n)} = \frac{1}{2\log^k(n)}$.

Next, we set $\mu(n) = n \geq 2\sqrt{n+1}$, $\epsilon = \frac{1}{2\log(n)}$, and $\kappa(\epsilon) \coloneqq 2^{-O\left(\log^3\left(\frac{1}{\epsilon}\right)\right)}$. For sufficiently large n we have that

$$\kappa(\epsilon) = \frac{1}{2^{O(\log^3(2\log^k(n)))}} \leq \frac{1}{2\log^k(n)} = \frac{n}{2n\log^k(n)} = \hat{f}(l)\mu(n).$$

Hence, we can apply Theorem 4.1 to f' to get a weighted threshold access structure over n parties g that is $\frac{1}{\log^k(n)}$-close to f' and admits an information-theoretic secret sharing scheme with share size $\tilde{O}\left(n\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$. Moreover, it holds that

$$\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)} = n^{\frac{O\left(\log^3\left(\frac{1}{\epsilon}\right)\right)}{\log(n)}} = n^{\frac{O(\log^3(2\log^k(n)))}{\log(n)}} = n^{o(1)},$$

which implies that the share size is $n^{1+o(1)}$. Finally, the triangle inequality implies that

$$\mathrm{dist}(f, g) \leq \mathrm{dist}(f, f') + \mathrm{dist}(f', g) \leq \frac{1}{2\log^k(n)} + \frac{1}{2\log^k(n)} = \frac{1}{\log^k(n)}.$$

□

4.2 Remarks on the Secret Sharing Techniques

At the end of the proof of Theorem 4.1 we have used Shamir's virtualization technique to construct our secret sharing scheme. Hence, one may wonder if the use of an alternative construction may lead to smaller shares. To answer this question, we move to combine our technique with the other existing proposals. First, we target the work of Benaloh and Leichter based on monotone formulae [BL88]. Later, we focus our attention on the work of Beimel and Weinreb using monotone circuits [BW06]. From there, we observe that the resulting schemes have larger share size than the one from Theorem 4.1. Finally, we present a brief discussion about lower bounds on information-theoretic secret sharing schemes for weighted threshold access structures.

Alternative Secret Sharing Schemes Constructions. Benaloh and Leichter [BL88] presented a secret sharing construction whose share size is linear in the size of any monotone formula realizing the access structure. In this regard, note that Theorem 2.7 states that any weighted threshold access structure has a monotone formula of polynomial size in the total weight. Therefore, given any weighted threshold access structure we can combine our approximation technique with these two results to obtain an information-theoretic secret sharing

scheme with total share size $O\left(\mu(n)^{10.6}\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$ for an ϵ-close weighted threshold access structure. More specifically, we first apply Theorem 3.1 to construct an ϵ-close weighted threshold access structure, then we use Theorem 2.7 to obtain a polynomial size monotone formula, and finally we construct the secret sharing scheme with polynomial share size.

The construction by Beimel and Weinreb [BW06] has two main steps. First, they describe logarithmic depth and polynomial size unbounded fan-in monotone circuits that compute any monotone weighted threshold function (Theorem 2.3). Later, they transform the circuit into a monotone boolean formula, that is transformed into a scheme by the technique mentioned above, obtaining the share size in Theorem 2.21. To construct these circuits they use the upper bound on the weights of $2^{\lceil n\log(n)\rceil}$ of Theorem 2.8. Hence, we can use our approximation technique to avoid using this bound. More in detail, we can apply Theorem 3.1 to construct an ϵ-close weighted threshold access structure and bound the weights by $\mu(n)\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}$. However, this strategy only leads to secret sharing schemes with share size $n^{O\left(\log\left(\log\left(\mu(n),\frac{1}{\epsilon}\right)\right)\right)}$, which is still quasipolynomial. This is because the quasipolynomial magnitude is due to the conversion from monotone circuits to monotone formulae, something in which our technique does not help.

On the Share Size Lower Bounds. The best lower bounds on the share size for weighted threshold access structures are far from the general ones. The fact that threshold access structures belong to this family gives a lower bound of $\Omega(\log n)$ on the share size. Moreover, there is a family of weighted threshold access structures that require information ratio close to 2 [FMBPV12]. These are the best known lower bounds.

A common technique for obtaining lower bounds on the share size for linear schemes is through counting arguments. In this regard, the next theorem gives upper and lower bounds on the number of weighted threshold access structures over n parties and shows that they correspond to a small subset within the set of monotone Boolean functions. Counting arguments like the ones in [KW93] give trivial bounds, in this case.

Theorem 4.4. *The number of weighted threshold access structures over n parties is $2^{\Theta(n^2)}$.*

Proof. First, notice that finding the number of weighted threshold access structures over n parties is equivalent to finding the number of monotone WTFs over n variables.

Now, let $\mathcal{T}$ (resp. $\mathcal{T}_M$) denote the set of all WTFs (resp. monotone WTFs) with n variables. In a previous work [Mur71], Muroga proved that

$$2^{\frac{n^2}{2}} \leq |\mathcal{T}| \leq 2^{n^2} \text{ for any } n.$$

Hence, to prove the theorem it suffices to show that $\frac{|\mathcal{T}|}{2^n} \leq |\mathcal{T}_M|$.

To do so, we define a surjective mapping $\varphi : \mathcal{T}'_M \to \mathcal{T}$, where $\mathcal{T}'_M$ is the set containing 2^n copies of each monotone LTF ordered from 0 to $2^n - 1$. In this setup, φ simply consists in taking the positive representation of any monotone LTF given by Remark 2.2 and mapping its k-th copy to the LTF obtained by negating the weights corresponding to the coordinates referred by the binary representation of k.

Hence, φ is surjective by construction because for any LTF with k weights smaller than zero we obtain a monotone LTF by taking the absolute value of these weights. In particular, its preimage consists in the k-th copy of some monotone LTF. □

4.3 Comparison with State-of-the-Art Proposals

We now compare our proposal with the state-of-the-art constructions. This is summarized in Fig. 1.

Share Size. Our construction and the ones from Benhamouda, Halevi, and Stambler are the unique schemes that offer polynomial share size. However, the share size of the works of from Benhamouda, Halevi, and Stambler also depend on the inverse of the gap $\beta - \alpha$, which leads to an increase of the share size when targeting ramp weighted threshold access structures with small gaps. With respect to the proposals from Shamir and Garg et al., their share size depends on the weights, which can be exponential in terms of the number of parties as stated in Theorem 2.8. For this reason, these proposals are more suitable for the cases in which there are lots of parties with small weights. Contrary to that, the scheme from Beimel and Weinreb fits well in weighted threshold access structures with high order weights because it has quasipolynomial share size in the number of parties.

Access Structure. The works from Shamir and Beimel and Weinreb are the only ones that construct secret sharing schemes for weighted threshold access structures, while the proposals from Garg et al. and Benhamouda, Halevi, and Stambler rely on the more flexible setting of ramp weighted threshold access structures. Concerning to this, note that the gap needed for the scheme of Garg et al. is significantly smaller, since the other gaps correspond to a fraction of the total weight W. In comparison, despite modifying the original weighted threshold access structure, our construction has the advantage that the resulting access structure is also a weighted threshold access structure whose error is $\frac{1}{\mathrm{polylog}(n)}$, which tends to zero as the number of parties tends to infinity. Furthermore, since our proposal has the error as an input parameter, we are able to tune the accuracy of the approximation as desired. In contrast, in the ramp proposals it is hard to establish a precise relation between the variation of the thresholds and the size of the gap.

Privacy and Linearity. Our proposal, the ones from Shamir, Beimel and Weinreb, and the rounding scheme from Benhamouda, Halevi, and Stambler have perfect privacy and are also linear. This makes them suitable for multiparty computation applications. The rest of the constructions only admit statistical secret sharing schemes and are not linear.

5 Computational Secret Sharing Schemes for Approximate Weighted Threshold Access Structures

In this section we construct computational secret sharing schemes for weighted threshold access structures with small share size. First, we introduce some auxiliary results necessary for our work. Later, we present the construction and show how to quantify the public information size with our approximation technique.

5.1 Succinct Computational Secret Sharing Schemes

In a recent work, Applebaum et al. [ABI$^+$23] construct computational secret sharing schemes with small share size for a wide set of access structures. More in detail, they introduce a new cryptographic primitive known as Projective Pseudorandom Generator (pPRG), show how to construct it from several assumptions such as RSA or the existence of OWF, and use it to obtain succinct computational secret sharing schemes, i.e. schemes whose share size is considerably small. We defer the definition of pPRG to Appendix B.2.

Their main result regarding pPRG is the following theorem.

Theorem 5.1 ([ABI$^+$23]). *Under the subexponential (resp. polynomial) RSA assumption, there exists a subexponential-robust pPRG (resp. polynomial-robust pPRG) with subexponential stretch (resp. arbitrary polynomial stretch) whose projective keys and public parameters are both strongly succinct, i.e. of length* $\log(m) \cdot \mathrm{poly}(\lambda)$*, where* m *is the output length and* λ *is the security parameter. The running time of generating the* m*-bit output of the pPRG is* $\tilde{O}(m) \cdot \mathrm{poly}(\lambda)$*.*

For our purposes, we require a slight generalization of the notion of pPRG known as block-pPRG, whose precise definition is presented in Appendix B.3. Block-pPRG are important because they are the building blocks for computational secret sharing schemes for monotone circuits with unbounded fan-in in which the share size is polylogarithmic in the number of gates. This is summarized in the following theorem.

Theorem 5.2 ([ABI$^+$23]). *Let* λ *be the security parameter. Assume that there is a robust block-pPRG in which the length of the projective keys is* $\log(m\lambda) \cdot \mathrm{poly}(\lambda)$ *and the length of the public parameters is* $\log(m\lambda) \cdot \mathrm{poly}(\lambda)$*, where* m *is the output length (number of blocks) of the generator and each block is of length* λ*. Then, there is a computational secret sharing scheme for monotone unbounded fan-in circuits whose share size is* $\log(m\lambda) \cdot \mathrm{poly}(\lambda)$ *and its public information size is* $\mathrm{poly}\left(\log\left(m\right), \lambda\right) + m_\wedge \lambda$*, where* m *is the number of gates and* $m_\wedge$ *is the number of AND-gates.*

5.2 Scheme Construction

A direct application of Theorem 5.1 and Theorem 5.2 to the polynomial size logarithmic depth monotone circuits for monotone WTF of Theorem 2.3 leads to the construction of a computational secret sharing scheme for weighted threshold access structures with polylogarithmic share size and public information of polynomial size in the number of parties. This is stated in the next theorem.

Theorem 5.3 (Theorem 1.4 restated). *Let λ be the security parameter. Under the subexponential RSA assumption, any weighted threshold access structure over n parties admits a computational secret sharing scheme where the size of the shares is $O\left(\text{poly}(\log(n), \lambda)\right)$ and the size of the public information is $O\left(\text{poly}(n, \lambda)\right)$.*

Proof. Given a weighted threshold access structure f, we use Theorem 2.3 to compute its polynomial size logarithmic depth monotone circuit. Then, we apply Theorem 5.1 and Theorem 5.2 to this circuit to obtain the desired computational secret sharing scheme.

The main drawback of the secret sharing scheme construction of Theorem 5.3 is that its public information size corresponds to the size of the monotone circuit, which is a polynomial of unknown degree. For this reason, it is worth trying to pin down the degree of that polynomial.

In order to do so, we combine our results on low-weight approximators for monotone LTFs with the work from Applebaum et al. In particular, we construct computational secret sharing schemes for approximate weighted threshold access structures that maintain the share size of Theorem 5.3 and whose public information size is a polynomial of concrete degree. The main statement follows.

Theorem 5.4. *Let λ be the security parameter, let $\kappa(\epsilon) = 2^{-O\left(\log^3\left(\frac{1}{\epsilon}\right)\right)}$ and let $\mu(n) \in \mathbb{R}$ be a function that satisfies $\mu(n) \geq 2\sqrt{n+1}$. Under the subexponential RSA assumption, for any weighted threshold access structure over n parties f and $\epsilon \in (0,1)$ with $\kappa(\epsilon) \leq \hat{f}(n)\mu(n)$, there exists a weighted threshold access structure over n parties that is ϵ-close to f admitting a computational secret sharing scheme where the size of the shares is $O\left(\text{poly}\left(\log\left(\mu(n), \frac{1}{\epsilon}\right), \lambda\right)\right)$ and the size of public information is $O\left(\mu(n)^{10.6}\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}, \text{poly}\left(\lambda\right)\right)$.*

Proof. The first part of the proof is analogue to that of Theorem 4.1, i.e. we use the switching lemmas (Appendix A.1) and Theorem 3.1 to obtain a weighted threshold access structure g with weight vector $\boldsymbol{v}$, threshold v_0, and $\text{dist}(f, g) < \epsilon$.

From there, instead of applying Shamir's virtualization technique, we use Theorem 2.7 to obtain a monotone formula for g of size $O\left(\mu(n)^{10.6}\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}\right)$. Now, since a monotone formula is indeed a monotone circuit, we can combine Theorem 5.1 and Theorem 5.2 to construct a computational secret sharing

scheme for g where the size of the shares is $O\left(\text{poly}\left(\log\left(\mu(n), \frac{1}{\epsilon}\right), \lambda\right)\right)$ and the size of public information is $O\left(\mu(n)^{10.6}\left(\frac{1}{\epsilon}\right)^{O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)}, \text{poly}\left(\lambda\right)\right)$. □

Now, we can mimic the approach done in the information-theoretic setting to obtain the following result as a consequence of Theorem 5.4.

Theorem 5.5. *Let $k \in \mathbb{N}$. Under the subexponential RSA assumption, for any weighted threshold access structure over n parties f, there exists a weighted threshold access structure over n parties $\frac{1}{\log^k(n)}$-close to f admitting a computational secret sharing scheme where the size of the shares is $O\left(\text{poly}\left(\log(n), \lambda\right)\right)$ and the size of the public information is $O\left(n^{10.6+o(1)}, \text{poly}\left(\lambda\right)\right)$.*

Proof. The proof follows the same steps as the proof of Theorem 4.3. The only difference is that we apply Theorem 5.4 to the intermediate weighted threshold access structure f' instead of Theorem 4.1. In this way, at the end of the procedure we obtain a computational secret sharing scheme for g with shares of size $O\left(\text{poly}\left(\log\left(n\right), \lambda\right)\right)$ and public information of size $O\left(n^{10.6+o(1)}, \text{poly}\left(\lambda\right)\right)$.

References

ABI+23. Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In *STOC 2023*, pages 1553–1566. ACM, 2023.

BCC+21. Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, and et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. Technical report, Chainlink Labs, 2021.

BD91. Ernest F. Brickell and Daniel M. Davenport. On the classification of ideal secret sharing schemes. *J. of Cryptology*, 4(73):123–134, 1991.

Bei11. Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology – Third International Workshop, IWCC 2011*, volume 6639 of *LNCS*, pages 11–46. Springer-Verlag, 2011.

BF20. Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. *IACR Cryptol. ePrint Arch.*, 2020:664, 2020.

BHS22. Fabrice Benhamouda, Shai Halevi, and Lev Stambler. Weighted secret sharing from wiretap channels. Cryptology ePrint Archive, Paper 2022/1578, 2022.

BHS23. Fabrice Benhamouda, Shai Halevi, and Lev Stambler. Weighted secret sharing from wiretap channels. In *4th Conference on Information-Theoretic Cryptography, ITC 2023*, volume 267 of *LIPIcs*, pages 8:1–8:19, 2023.

BL88. Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shaffi Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 27–35. Springer-Verlag, 1988.

Bla79. George Robert Blakley. Safeguarding cryptographic keys. In *Proc. of the 1979 AFIPS National Computer Conference*, volume 48 of *AFIPS Conference proceedings*, pages 313–317. AFIPS Press, 1979.

Bri89. Ernest F. Brickell. Some ideal secret sharing schemes. *Journal of Combin. Math. and Combin. Comput.*, 6:105–113, 1989.

BTW08. Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. *SIAM J. Discrete Math.*, 22(1):360–397, 2008.

BW06. Amos Beimel and E. Weinreb. Monotone circuits for monotone weighted threshold functions. *Inform. Process. Lett.*, 97(1):12–18, 2006. Conference version: *Proc. of 20th Annu. IEEE Conf. on Computational Complexity*, pages 67-75, 2005.

Cho61. C. K. Chow. On the characterization of threshold functions. In *2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, pages 34–38, 1961.

DDFS14. Anindya De, Ilias Diakonikolas, Vitaly Feldman, and Rocco A. Servedio. Nearly optimal solutions for the chow parameters problem and low-weight approximation of halfspaces. *J. ACM*, 61(2), apr 2014.

DPTX24. Sourav Das, Benny Pinkas, Alin Tomescu, and Zhuolun Xiang. Distributed randomness using weighted vrfs. Cryptology ePrint Archive, Paper 2024/198, 2024. `https://eprint.iacr.org/2024/198`.

FMBPV12. O. Farràs, J. R. Metcalf-Burton, C. Padró, and L. Vázquez. On the optimization of bipartite secret sharing schemes. *Des. Codes Cryptogr.*, 63:255–271, 2012.

FP12. Oriol Farràs and Carles Padró. Ideal hierarchical secret sharing schemes. *IEEE Transactions on Information Theory*, 58(5):3273–3286, 2012.

GJM+23. Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Cryptography with weights: Mpc, encryption and signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 295–327, Cham, 2023. Springer Nature Switzerland.

Hås94. J. Håstad. On the size of weights for threshold gates. *SIAM J. on Discrete Mathematics*, 7(3):484–492, 1994.

KGH83. Ehud D. Karnin, Jonathan W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Trans. on Information Theory*, 29(1):35–41, 1983.

KRDO17. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham, 2017. Springer International Publishing.

KW93. Mauricio Karchmer and Avi Wigderson. On span programs. In *8th Structure in Complexity Theory*, pages 102–111, 1993.

LV18. Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *50th STOC*, pages 699–708, 2018.

Mig83. Maurice Mignotte. How to share a secret. In Thomas Beth, editor, *Cryptography*, pages 371–375, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.

Mo23. Songbao Mo. Ideal hierarchical secret sharing and lattice path matroids. *Des. Codes Cryptogr.*, 91(4):1335–1349, 2023.

MPSV99. P. Morillo, C. Padró, G. Sáez, and J. L. Villar. Weighted threshold secret sharing schemes. *Inform. Process. Lett.*, 70(5):211–216, 1999.

Mur71. S. Muroga. *Threshold Logic and Its Applications*. Wiley-Interscience, 1971.

O'D14. Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, USA, 2014.

OS11. Ryan O'Donnell and Rocco A. Servedio. The chow parameters problem. *SIAM Journal on Computing*, 40(1):165–199, 2011.

PS00. Carles Padró and Germán Sáez. Secret sharing schemes with bipartite access structure. *IEEE Transactions on Information Theory*, 46(7):2596–2604, 2000.

SB91. Kai-Yeung Siu and Jehoshua Bruck. On the power of threshold circuits with small weights. *SIAM Journal on Discrete Mathematics*, 4(3):423–435, 1991.

Ser04. R. Servedio. Monotone Boolean formulas can approximate monotone linear threshold functions. *Discrete Appl. Math.*, 142(1-3):181–187, 2004.

Ser06. R.A. Servedio. Every linear threshold function has a low-weight approximator. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 18–32, 2006.

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

Sim88. Gustavus J. Simmons. How to (really) share a secret. In *CRYPTO*, pages 390–448, 1988.

Tas07. T. Tassa. Hierarchical threshold secret sharing. *J. of Cryptology*, 20(2):237–264, 2007. Conference version in *Proc. of the First Theory of Cryptography Conference – TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 473–490. Springer-Verlag, 2004.

TD09. Tamir Tassa and Nira Dyn. Multipartite secret sharing by bivariate interpolation. *J. Cryptology*, 22(2):227–258, 2009.

A Statements and Proofs of Results in Section 2

A.1 Switching lemmas

The next lemma shows that using ϕ we can switch from any monotone WTF to an equivalent monotone LTF without modifying the weight of any coordinate.

Lemma A.1. *For any $\boldsymbol{w} \in \mathbb{R}_+^n$ and $\sigma \in \mathbb{R}$ let f be the monotone WTF given by $f(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$ and let $W(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$. Let $\boldsymbol{\alpha} \in \{0,1\}^n$ be a forbidden vector of f with maximum weight and let g be the monotone LTF given by $g(\boldsymbol{y}) = \mathrm{LTF}(\boldsymbol{w}, W(\overline{\boldsymbol{\alpha}}) - W(\boldsymbol{\alpha}))$. It holds that $f(\boldsymbol{x}) = 1$ if and only if $g(\phi(\boldsymbol{x})) = -1$.*

Proof. First of all, notice that for any $\boldsymbol{x} \in \{0,1\}^n$ we have that $W(\boldsymbol{x}) + W(\overline{\boldsymbol{x}}) = W(\mathbf{1}^n)$, i.e. the sum of the weights of a vector and its complementary are equal to the total weight.

For any $\boldsymbol{x} \in \{0,1\}^n$, having $f(\boldsymbol{x}) = 1$ is clearly equivalent to imposing that $W(\boldsymbol{x}) \geq \sigma$. Moreover, by definition we have that $\sigma > W(\boldsymbol{\alpha})$. Hence, combining both inequalities and the previous comment we have that $f(\boldsymbol{x}) = 1$ if and only if $W(\overline{\boldsymbol{\alpha}}) - W(\boldsymbol{\alpha}) > W(\overline{\boldsymbol{x}}) - W(\boldsymbol{x})$, which by definition is the same as stating that $g(\phi(\boldsymbol{x})) = -1$. □

The lemma that follows corresponds to the converse of Lemma A.1, i.e. it shows that using ϕ we can pass from any monotone LTF to an equivalent monotone WTF without modifying the weight of any coordinate.

Lemma A.2. *For any $\boldsymbol{w} \in \mathbb{R}_+^n$ and $\sigma \in \mathbb{R}$ let f be the monotone LTF given by $f(\boldsymbol{x}) = \mathrm{LTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$ and let $W(\boldsymbol{x}) = \sum_{x_i=1} w_i$. Let $\boldsymbol{\alpha} \in \{-1,1\}^n$ be a forbidden vector of f with maximum weight and let g be the monotone WTF given by $g(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{w}, W(\overline{\boldsymbol{\alpha}}))$. It holds that $f(\boldsymbol{x}) = 1$ if and only if $g(\phi^{-1}(\boldsymbol{x})) = 0$.*

Proof. First of all, notice that for any $\boldsymbol{x} \in \{-1,1\}^n$ we have that $W(\boldsymbol{x}) + W(\overline{\boldsymbol{x}}) = W(\mathbf{1}^n)$.

For any $\boldsymbol{x} \in \{-1,1\}^n$, having $f(\boldsymbol{x}) = 1$ is clearly equivalent to imposing that $W(\boldsymbol{x}) - W(\overline{\boldsymbol{x}}) \geq \sigma$. Moreover, by definition we have that $\sigma > W(\boldsymbol{\alpha}) - W(\overline{\boldsymbol{\alpha}})$. Hence, combining both inequalities and the previous comment we have that $f(\boldsymbol{x}) = 1$ if and only if $W(\overline{\boldsymbol{\alpha}}) > W(\overline{\boldsymbol{x}})$, which by definition is the same as stating that $g(\phi^{-1}(\boldsymbol{x})) = 0$. □

A.2 Proof of Lemma 2.13

Proof. First, we prove that $\hat{f}(i) \geq \hat{g}(i)$. By definition, we must prove that $\mathbf{E}[f(\boldsymbol{x})x_i] \geq \mathbf{E}[P_1(f(\boldsymbol{x}))x_i]$. Since $|P_1(f(\boldsymbol{x}))| \leq 1$, it suffices to prove that whenever there exists $\boldsymbol{x} \in \{-1,1\}^n$ such that $f(\boldsymbol{x})x_i \leq -1$, then there exists $\boldsymbol{y} \in \{-1,1\}^n$ such that $f(\boldsymbol{y})y_i \geq -f(\boldsymbol{x})x_i$.

In this regard, if $f(\boldsymbol{x})x_i \leq -1$ it holds that either $f(\boldsymbol{x}) < -1$ and $x_i = 1$ or $f(\boldsymbol{x}) > 1$ and $x_i = -1$. In both cases, by monotonicity if we define $\boldsymbol{y} = \boldsymbol{x}^{\oplus i}$ we have that $f(\boldsymbol{y})y_i \geq |f(\boldsymbol{x})x_i|$ as desired.

Now, to prove that $\hat{g}(i) \geq 0$, it suffices to notice that by construction if f is monotone then g is monotone too and then to use again the monotonicity argument for the values $\boldsymbol{x} \in \{-1,1\}^n$ such that $g(\boldsymbol{x})x_i \leq 0$. □

B Computational Secret Sharing Schemes

In this section, we provide the full definitions of computational secret sharing schemes and projective pseudorandom generators. All these definitions are from [ABI+23], and we provide them for the sake of completeness.

B.1 Security of Computational Secret Sharing Schemes

Definition B.1 (Security). *Let $\mathcal{S} = \{0,1\}$, let $\lambda \in \mathbb{N}$ be the security parameter, let f be an access structure over n parties, and let* (Share, Reconstruct) *be a computational secret sharing scheme for f. Consider the following game between a non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and a challenger:*

1. *The adversary $\mathcal{A}$ on input $\mathbf{1}^\lambda$ chooses a forbidden set A and sends it to the challenger.*

2. *The challenger chooses a uniformly random secret $s \in \mathcal{S}$. It computes* $\mathsf{Share}(s)$ *and sends* $\mathsf{Share}(s)_A$ *to the adversary.*
3. *The adversary outputs a value $s' \in \mathcal{S}$.*

The adversary wins the game if $s' = s$.

We say that the computational secret sharing scheme is $t(\lambda)$-secure if for every non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and sufficiently large λ, the probability that $\mathcal{A}$ wins is at most $\frac{1}{2} + \frac{1}{t(\lambda)}$. By default, we require $t(\lambda)$-security for every polynomial $t(\cdot)$. In any case, we always assume that $t > \lambda$.

B.2 Projective Pseudorandom Generators

Definition B.2 (Projective Pseudorandom Generator). *A projective Pseudorandom Generator (pPRG) is a triple of algorithms* $\mathsf{pPRG} = (\mathsf{pPRG.Setup}; \mathsf{pPRG.KeyGen}; \mathsf{pPRG.Eval})$ *with the following syntax:*

- ***Setup***: $\mathsf{pPRG.Setup}(1^\lambda, 1^m) \rightarrow$*(params, msk) is a randomized poly-time algorithm that takes as input a security parameter λ and an output length parameter m, and samples public parameters params and master secret key msk. We assume that the public parameters are of length at least λ and that one can recover in time poly(λ, $\log m$) the values of the security parameter and the output length m from params.*
- ***Key Generation***: $\mathsf{pPRG.KeyGen}$*(params, msk, T) is a deterministic poly-time algorithm that takes as input the public parameters params, a secret key msk, and a set $T \subseteq [m]$ represented by its m-bit characteristic vector, and outputs a projective key $a\{T\}$.*
- ***Evaluation:*** $\mathsf{pPRG.Eval}$*(params, msk, T) is a deterministic poly-time algorithm that takes as input the public parameters params, a projective key $a\{T\}$, and a set $T \subseteq [m]$ represented by an m-bit characteristic vector, and outputs a string $y \in \{0,1\}^{|T|}$. We refer to c as the string that is the output of the pPRG.*

Since the description length of params (resp. T) is at least λ (resp., the output length m), the algorithms $\mathsf{pPRG.KeyGen}$ *and* $\mathsf{pPRG.Eval}$ *are implicitly allowed to run in time poly(λ, m). We require the following properties:*

- ***Correctness:*** *Correctness requires that for every λ, m, (params, msk) $\in$* $\mathsf{pPRG.Setup}(1^\lambda, 1^m)$*, $T \subseteq [m]$, and $a\{T\} =$* $\mathsf{pPRG.KeyGen}$*(params, msk, T), it holds that $y = c[T]$, where $y =$* $\mathsf{pPRG.Eval}$*(params, $a\{T\}$, T) is the string generated by the T-projective key $a\{T\}$ and $c =$* $\mathsf{pPRG.Eval}$*(params, msk) is the string generated by (params, msk).*
- ***Succinctness:*** *Weak succinctness requires that there is a constant $\delta > 0$ such that for every $T \subseteq [m]$ the bit-length of $a\{T\}$ is $m^{1-\delta}$ poly(λ). Strong succinctness requires that there is a fixed polynomial p such that the size of $a\{T\}$ is $p(\log m, \lambda)$.*
- ***Security:*** *Consider the following game between an adversary $\mathcal{A}$ and a challenger:*

1. *Given an input* 1^λ*, the adversary* $\mathcal{A}$ *chooses* 1^m *and* $T \subseteq [m]$ *and sends* $(1^m, T)$ *to the challenger.*
2. *The challenger samples* $(params, msk) \leftarrow \mathsf{pPRG.Setup}(1^\lambda, 1^m)$*, computes* $c \leftarrow \mathsf{pPRG.Eval}(params, msk)$*, sets* $c_1 \leftarrow c[\overline{T}]$*, and samples* c_0 *uniformly from* $\{0,1\}^{|\overline{T}|}$*, where* $\overline{T} = [m] \setminus T$ *is the complement of* T*. The challenger samples* b *uniformly from* $\{0,1\}$ *and sends params,* $a\{T\} = \mathsf{pPRG.KeyGen}(params, msk, T)$*, and* c_b *to the adversary.*
3. *The adversary* $\mathcal{A}$ *outputs a bit* b_0 *and wins if* $b = b_0$*.*

We say that pPRG is $t(\lambda)$*-secure if for every non-uniform* $t(\lambda)$*-time adversary* $\mathcal{A}$ *the probability that* $\mathcal{A}$ *wins is at most* $\frac{1}{2} + \frac{1}{t(\lambda)}$*.*

An additional feature of a projective PRG is robustness, a strengthening of the above security definition.

- ***Robustness:*** *Consider the following game between an adversary* $\mathcal{A}$ *and a challenger:*
  1. *Given an input* 1^λ*, the adversary* $\mathcal{A}$ *chooses* 1^m *and sets* $T_1, \ldots, T_l \subseteq [m]$ *and sends them to the challenger. Let* $T = T_1 \cup \ldots \cup T_l$ *denote the union of these sets.*
  2. *The challenger samples* $(params, msk) \leftarrow \mathsf{pPRG.Setup}(1^\lambda, 1^m)$*, computes* $c \leftarrow \mathsf{pPRG.Eval}(params, msk)$*, sets* $c_1 \leftarrow c[\overline{T}]$*, and samples* c_0 *uniformly from* $\{0,1\}^{|\overline{T}|}$*, where* $\overline{T} = [m] \setminus T$ *is the complement of* T*. Furthermore, the challenger samples* b *uniformly from* $\{0,1\}$ *and sends params,* $a\{T_i\} = \mathsf{pPRG.KeyGen}(params, msk, T_i)$ *for all* $i \in [l]$*, and* c_b *to the adversary.*
  3. *The adversary* $\mathcal{A}$ *outputs a bit* b' *and wins if* $b' = b$*.*

  We say that pPRG is $t(\lambda)$*-robust if for every non-uniform* $t(\lambda)$*-time adversary* $\mathcal{A}$ *the probability that* $\mathcal{A}$ *wins is at most* $\frac{1}{2} + \frac{1}{t(\lambda)}$*.*

B.3 Block-Projective Pseudorandom Generators

Definition B.3 (Block-Projective Pseudorandom Generator). *A Block-Projective Pseudorandom Generator (block-pPRG) is a natural generalization of pPRG. Specifically, in a block pPRG we think of the pseudorandom output as a sequence* $(c_1, \ldots, c_m)$ *of* m *blocks, where each block* c_i *is of length* λ*. A projective key* $a\{T\}$ *of a set* $T \subseteq [m]$ *should allow computing all the blocks* c_i *for which* $i \in T$*. The security and robustness games are defined naturally where the only difference is that* c_0 *is sampled uniformly from* $(\{0,1\}^\lambda)^{\overline{T}}$*.*

Remark B.4. Any (robust) pPRG with an output length of $m' = m\lambda$ can be viewed as a (robust) block pPRG of length m by parsing the outputs as blocks and by setting the projective key of T to $a\{T'\}$, where T' is the set of all location that fall inside the blocks whose index is in T. Alternatively, one can simply concatenate λ-independent copies of a pPRG of output length m and set the i-th bit of the j-th block to be the j-th output bit of the i-th copy. This transformation preserves robustness and increases the key size and the size of

the public parameters by a factor of λ, and therefore succinctness and strong succinctness are preserved. Finally, we note that concrete constructions (e.g. those based on RSA) can be easily modified to obtain block-pPRGs directly at a minor cost.

C Proofs of Results in Section 3

C.1 Proof of Proposition 3.3

Proof. First, notice that whenever $\|\boldsymbol{\chi}_f - \boldsymbol{\chi}_{\tilde{g}}\| \leq 2\epsilon$ the triangle inequality implies that

$$\|\boldsymbol{\chi}_f - \boldsymbol{\chi}_g\| \leq \|\boldsymbol{\chi}_f - \boldsymbol{\chi}_{\tilde{g}}\| + \|\boldsymbol{\chi}_{\tilde{g}} - \boldsymbol{\chi}_g\| \leq 2\epsilon + \sqrt{\sum_{i=0}^{n} \left(\frac{\epsilon}{\mu(n)}\right)^2} \leq 2\epsilon + \frac{\epsilon}{2} < 3\epsilon.$$

Next, we prove that the output g is a monotone LBF. Let α_i be the i-th coefficient of g' for any $i \in [n]$, i.e. $g'(\boldsymbol{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$. By definition, $g = P_1(g')$, so to prove that g is monotone it suffices to prove the following claim.

Claim. α_i is non-negative for any $i \in [n]$.

Proof of the Claim. The construction of the Chow parameters implies that $\alpha_i = \hat{g}'(i)$ for any $i \in [n]$. Hence, the claim is equivalent to require that all the Chow parameters of g' are non-negative except for the first one.[6]

Let g_t and g'_t denote the instances of the functions g and g' after the t-th iteration of the loop of ApproximateLBF (Algorithm 1, steps 4-8), and let $h_t = \sum_{i=0}^{n} (\hat{f}(i) - \tilde{g}_t(i)) x_i$. Instead of proving the claim, we prove the following stronger statement: at any step t, $\hat{g}'_t(i) \geq 0$ for any $i \in [n]$.

We prove it by induction on t. For $t = 0$ the result follows trivially for any $i \in [n]$ because $g'_0 \equiv 0$.

We now suppose it is true for t and prove it for $t + 1$. By construction, we have that

$$\begin{aligned}\hat{g}'_{t+1}(i) = \hat{g}'_t(i) + \frac{1}{2} h_t(i) = \hat{g}'_t(i) + \frac{1}{2}(\hat{f}(i) - \tilde{g}_t(i)) \geq \\ \geq \hat{g}'_t(i) + \frac{1}{2}(\hat{f}(i) - \hat{g}'_t(i) - \frac{\epsilon}{\mu(n)}) \geq \frac{1}{2}(\hat{f}(i) + \hat{g}'_t(i) - \frac{\epsilon}{\mu(n)}) \geq 0,\end{aligned}$$

where the first inequality holds because $\tilde{g}_t(i) \leq \hat{g}_t(i) + \frac{\epsilon}{\mu(n)}$, the second inequality follows from g'_t being monotone by hypothesis and applying Lemma 2.13, and the last inequality is derived using that by hypothesis $\hat{g}'_t(i) \geq 0$ and $\hat{f}(i) \geq \frac{\epsilon}{\mu(n)}$.

This concludes the proof of the claim. □

[6] Notice that to ensure monotonicity of a LTF the sign of the threshold does not matter. Hence, we do not require its 0-th Chow parameter to be non-negative.

It only remains to prove the probabilistic nature of the result. This is a technical requirement to obtain the desired running time. In more detail, the running time of the algorithm is mainly determined by computing the Chow parameters $\boldsymbol{\chi}_g$ at each iteration (step 7), which requires exponential running time for total precision. However, since we only require the Chow parameters to be computed with precision $\frac{\epsilon}{2\mu(n)}$, we can use the Chernoff bounds to obtain a probabilistic but more efficient result. In particular, with probability at least $1-\delta$, we obtain the Chow parameters $\boldsymbol{\chi}_g$ with precision $\frac{\epsilon}{2\mu(n)}$ by using the empirical mean of $g(\boldsymbol{x})x_i$ on $O\left(\frac{\mu(n)^2}{\epsilon^2}\log\left(\frac{n}{\epsilon\delta}\right)\right)$ random points as our estimate of $\hat{g}(i)$ for any $i=0,\ldots,n$. □

C.2 Proof of Proposition 3.4

Proof. To prove that the main loop of the algorithm terminates, we define a potential function at step t as

$$E(t) = \mathbf{E}[(f-g_t)^2] + 2\mathbf{E}[(f-g_t)(g_t-g'_t)] = \mathbf{E}[(f-g_t)(f-2g'_t+g_t)].$$

We next proof the following claim about the potential function $E(t)$.

Claim. $E(t+1)-E(t) \leq -\epsilon^2$.

Proof. By construction, we have that

$$\begin{aligned} E(t+1)-E(t) &= \mathbf{E}[(f-g_{t+1})(f-2g'_{t+1}+g_{t+1})] - \mathbf{E}[(f-g_t)(f-2g'_t+g_t)] = \\ &= \mathbf{E}[(f-g_t)(2g'_t-2g'_{t+1}) + (g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})] = \\ &= \mathbf{E}[(f-g_t)h_t] + \mathbf{E}[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})]. \quad (1) \end{aligned}$$

Hence, to bound $E(t+1)-E(t)$ it suffices to give upper bounds on $\mathbf{E}[(f-g_t)h_t]$ and $\mathbf{E}[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})]$ independently.

We first prove that

$$\mathbf{E}[(f-g_t)h_t] \geq \rho^2 - \frac{\rho\epsilon}{2} \quad (2)$$

To prove Equation (2) it suffices to use the Cauchy-Schwartz inequality and Theorem 2.10 to get that

$$\begin{aligned} \mathbf{E}[(f-g_t)h_t] &= \sum_{i=0}^{n}(\hat{f}(i)-\hat{g}_t(i))(\hat{f}(i)-\tilde{g}_t(i)) = \\ &= \sum_{i=0}^{n}(\hat{f}(i)-\tilde{g}_t(i))(\tilde{g}_t(i)-\hat{g}_t(i)) + \sum_{i=0}^{n}(\hat{f}(i)-\tilde{g}_t(i))^2 \geq \rho^2 - \frac{\rho\epsilon}{2}. \end{aligned}$$

Next, to upper bound the expression $\mathbf{E}[(g_{t+1}-g_t)(2g'_{t+1}-g_t-g_{t+1})]$ we prove that for every $\boldsymbol{x}\in\{-1,1\}^n$

$$(g_{t+1}(\boldsymbol{x})-g_t(\boldsymbol{x}))(2g'_{t+1}(\boldsymbol{x})-g_t(\boldsymbol{x})-g_{t+1}(\boldsymbol{x})) \leq \frac{h_t(\boldsymbol{x})^2}{2}.$$

We first observe that

$$|g_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x})| = \left| P_1\left(g'_t(\boldsymbol{x}) + \frac{h_t(\boldsymbol{x})}{2}\right) - P_1(g'_t(\boldsymbol{x}))\right| \leq \left|\frac{h_t(\boldsymbol{x})^2}{2}\right|$$

because a projection operation does not increase the distance.

Second, we prove that

$$|2g'_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x}) - g_{t+1}(\boldsymbol{x})| \leq |g'_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x})| + |g'_{t+1}(\boldsymbol{x}) - g_{t+1}(\boldsymbol{x})|.$$

This is because

$$|g'_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x})| = \left|\frac{h_t(\boldsymbol{x})}{2} + g'_t(\boldsymbol{x}) - g_t(\boldsymbol{x})\right| \leq \left|\frac{h_t(\boldsymbol{x})}{2}\right|$$

unless $g'_t(\boldsymbol{x}) - g_t(\boldsymbol{x}) \neq 0$ and $g'_t(\boldsymbol{x}) - g_t(\boldsymbol{x})$ has the same sign as $h_t(\boldsymbol{x})$. In that case, the definition of P_1 implies that

$$|g_t(\boldsymbol{x})| = \text{sign}(g'_t(\boldsymbol{x})) \text{ and } \text{sign}(h_t(\boldsymbol{x})) = \text{sign}(g'_t(\boldsymbol{x}) - g_t(\boldsymbol{x})) = g_t(\boldsymbol{x}).$$

However, this means that

$$|g'_{t+1}(\boldsymbol{x})| \geq |g'_t(\boldsymbol{x})| > 1 \text{ and } \text{sign}(g'_{t+1}(\boldsymbol{x})) = \text{sign}(g'_t(\boldsymbol{x})) = g_t(\boldsymbol{x}).$$

As a result

$$g_{t+1}(\boldsymbol{x}) = g_t(\boldsymbol{x}) \text{ and } (g_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x}))(2g'_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x}) - g_{t+1}(\boldsymbol{x})) = 0.$$

Similarly, for the second part we have that if

$$|g'_{t+1}(\boldsymbol{x}) - g_{t+1}(\boldsymbol{x})| \geq \left|\frac{h_t(\boldsymbol{x})}{2}\right|$$

then

$$g_{t+1}(\boldsymbol{x}) = \text{sign}(g'_{t+1}(\boldsymbol{x})) \text{ and } |g'_{t+1}(\boldsymbol{x})| \geq \left|\frac{h_t(\boldsymbol{x})}{2}\right| + 1.$$

This implies that

$$|g'_t(\boldsymbol{x})| \geq |g'_{t+1}(\boldsymbol{x})| - \left|\frac{h_t(\boldsymbol{x})}{2}\right| > 1$$

and

$$g_t(\boldsymbol{x}) = \text{sign}(g'_t(\boldsymbol{x})) = \text{sign}(g'_{t+1}(\boldsymbol{x})) = g_{t+1}(\boldsymbol{x}).$$

Altogether, we obtain that

$$(g_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x}))(2g'_{t+1}(\boldsymbol{x}) - g_t(\boldsymbol{x}) - g_{t+1}(\boldsymbol{x})) \leq \frac{h_t(\boldsymbol{x})^2}{2}.$$

Hence, applying Theorem 2.10 we get that

$$\mathbf{E}[(g_{t+1} - g_t)(2g'_{t+1} - g_t - g_{t+1})] \leq \mathbf{E}[h_t^2] = \frac{\rho^2}{2}, \tag{3}$$

which gives an upper bound of $\mathbf{E}[(g_{t+1} - g_t)(2g'_{t+1} - g_t - g_{t+1})]$.

Finally, by substituting Equation (2) and Equation (3) into Equation (1), we obtain the claimed decrease in the potential function

$$E(t+1) - E(t) \leq -\rho^2 + \frac{\epsilon\rho}{2} + \frac{\rho^2}{2} = -\frac{\rho}{2}(\rho - \epsilon) \leq -\epsilon^2,$$

where in the last inequality we have used that $\rho > 2\epsilon$. □

Once the claim is proved, we notice that for all t

$$E(t) = \mathbf{E}[(f - g_t)^2] + 2\mathbf{E}[(f - g_t)(g_t - g'_t)] \geq 0.$$

This holds because for every $\boldsymbol{x} \in \{-1, 1\}^n$ if $g_t(\boldsymbol{x}) - g'_t(\boldsymbol{x})$ is non-zero, by the definition of P_1 we have that

$$g_t(\boldsymbol{x}) = \text{sign}(g'_t(\boldsymbol{x})) \text{ and } \text{sign}(g_t(\boldsymbol{x}) - g'_t(\boldsymbol{x})) = -g_t(\boldsymbol{x}).$$

In this case,

$$f(\boldsymbol{x}) - g(\boldsymbol{x}) = 0 \text{ or } \text{sign}(f(\boldsymbol{x}) - g_t(\boldsymbol{x})) = -g_t(\boldsymbol{x})$$

and hence

$$(f(\boldsymbol{x}) - g_t(\boldsymbol{x}))(g_t(\boldsymbol{x}) - g'_t(\boldsymbol{x})) \leq 0.$$

Therefore,

$$\mathbf{E}[(f - g_t)(g_t - g'_t)] \leq 0 \text{ and clearly } \mathbf{E}[(f - g_t)^2] \leq 0.$$

By construction, it is clear that $E(0) = 1$ and therefore the process will stop after at most $\frac{1}{\epsilon^2}$ steps. □

C.3 Proof of Proposition 3.5

Proof. We start by proving the bound on the weights. Let T denote the number of iterations of the algorithm. By our construction, the function $g_T = P_1(\sum_{t \leq T} h_t)$ is a LBF represented by weight vector w such that $w_i = \sum_{j \leq T} \frac{1}{2}(\hat{f}(i) - \tilde{g}_j(i))$.

Our roundings of the estimates of Chow parameters of g_t ensure that each of $\frac{1}{2}(\hat{f}(i) - \tilde{g}_j(i))$ is a multiple of $k = \frac{\epsilon}{2\mu(n)}$. Hence, $g_t = \text{LBF}(k\boldsymbol{v}, kv_0)$, where the vector $\boldsymbol{v}$ has only integer components.

At every step j, we have that

$$\sqrt{\sum_{i=0}^{n} (\hat{f}(i) - \tilde{g_j}(i))^2} \leq \left\|\boldsymbol{\chi}_f - \boldsymbol{\chi}_{g_j}\right\| + \left\|\boldsymbol{\chi}_{g_j} - \boldsymbol{\chi}_{\tilde{g}_j}\right\| \leq 2\sqrt{\text{dist}(f, g)} + \frac{\epsilon}{2}$$
$$\leq 4 + \frac{\epsilon}{2} = O(1),$$

where the first inequality is due to the triangle inequality, the second inequality follows from Theorem 2.16, and the last one is because $\text{dist}(f, g) \leq 2$.

Therefore, since there are at most ϵ^{-2} steps, the triangle inequality implies that $\|\boldsymbol{w}\| = O(\epsilon^{-2})$ and hence $\|\boldsymbol{v}\| = \frac{\|\boldsymbol{w}\|}{k} = O\left(\frac{\mu(n)}{\epsilon^3}\right)$.

The running time of the algorithm is essentially determined by finding $\boldsymbol{\chi}_{\tilde{g}_t}$ in each step t. As discussed in the proof of Proposition 3.3, to ensure that with probability $1-\delta$ all the estimates of $\boldsymbol{\chi}_{g_t}$ are within $\frac{\epsilon}{2\mu(n)}$ of the true values requires evaluating each g_t on $O\left(\left(\frac{\mu(n)^2}{\epsilon^2}\right)\log\left(\frac{n}{\epsilon\delta}\right)\right)$ random points. Moreover, evaluating g_t on any point $\boldsymbol{x} \in \{-1,1\}^n$ takes $O(n)$ time and recall that there is a total of $O(\epsilon^{-2})$ steps. Therefore, combining all these bounds we get the claimed total running time. □