# Revisiting Beimel-Weinreb Weighted Threshold Secret Sharing Schemes

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 secret sharing scheme, 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, Beimel and Weinreb [IPL 2006] presented the best general constructions: The scheme with computational security has a total share size polynomial in $n$, while the scheme with perfect security has a total share size $n^{O(\log n)}$. However, these constructions require the use of shallow monotone sorting networks, which limits their practical use.

In this work, we revisit weighted threshold secret sharing from a circuit-based perspective. By considering alternative circuits and formulas that avoid monotone sorting networks, we obtain substantial improvements in two directions.

First, in the computational setting, we provide a computational scheme that is feasible in practice. Assuming the existence of one-way functions with security parameter $\lambda$, we show that any weighted threshold access structure over $n$ parties with threshold $\sigma$ admits a computational secret sharing scheme with share size $\lambda$, public information of size $O(\lambda n^2 \log \sigma)$, and a reconstruction procedure in which any authorized subset needs to download only $O(\lambda n \log \sigma)$ bits of public information. Notably, for weight distributions that arise in the most widely deployed blockchain networks, our construction reduces the total share size ranging from 300× to 6700× compared to the best previously known schemes.

Second, extending these techniques, we obtain improved information-theoretic ramp weighted threshold secret sharing schemes. For any weights, threshold $\sigma$, and $\epsilon < 1$, we construct a $(\sigma, (1+\epsilon)\sigma)$-ramp weighted threshold scheme with share size $O\left((n/\epsilon) \log n\right)$, reducing the share size with respect to the state-of-the-art solutions.

**Keywords:** Secret sharing scheme · weighted threshold access structure · threshold cryptography

## 1 Introduction

A secret sharing scheme is a cryptographic primitive that allows a dealer to share a secret among a set of parties in such a way that only certain subsets of them, called authorized, can recover the secret. The access structure of a scheme is the collection of its authorized subsets. Secret sharing schemes were introduced independently by Shamir [Sha79] and Blakley [Bla79] in 1979. They presented schemes for threshold access structures, i.e., schemes where the authorized subsets are those whose size is at least a given threshold.

Traditional secret sharing schemes provide information-theoretic security, remaining secure in the presence of adversaries with unbounded computational power; in contrast, computational secret sharing schemes rely on cryptographic assumptions to ensure security. This security reduction is motivated by the possibility of constructing more efficient schemes [Yao89,Kra94,ABI+23].

Secret sharing schemes are used as a building block in many cryptographic protocols, including secure multiparty computation and threshold encryption (see [Bei25]). In these contexts, threshold schemes are especially convenient and widely used. Nevertheless, there are situations that require schemes for non-threshold access structures.

*Weighted threshold access structures* (WTASs) can be seen as a natural generalization of the threshold ones. In a WTAS, each party is assigned a weight according to its importance, and a subset is authorized if the total weight of its parties meets or exceeds a specified threshold. WTASs are particularly appropriate in situations where parties have varying levels of significance. For example, in the proof-of-stake model, each validator's stake depends on the amount of cryptocurrency they own, and their influence is proportional to their stake [KRDO17,BCC+21]. These scenarios highlight the need for secret sharing schemes for WTASs.

Despite their utility, constructing efficient secret sharing schemes for WTASs poses several challenges. For these access structures, the share size of the best known information-theoretic schemes is either linear in the weights [Sha79] or quasipolynomial in the number of parties [BW06], often resulting in long shares. On the computational side, Beimel and Weinreb [BW06] presented a scheme with polynomial share size. However, it requires the use of shallow monotone sorting networks, whose practical applicability is limited by their intricate constructions and the large constants arising in their concrete complexity [AKS83,Pat90,Chv92]. These limitations not only increase the share size of their scheme, but also make it difficult to provide precise upper bounds on the share size.

Recently, several works have addressed these efficiency limitations by scaling down weights and relaxing both the privacy and correctness requirements [GJM+23,BHS23,TF24,FG24], leading to the notion of *ramp WTASs*. In this setting, and each party has a single weight, but there can be a gap between the privacy and correctness thresholds. This relaxed model enables significantly smaller share sizes at the cost of weaker privacy and correctness guarantees.

### 1.1 Our Results

In this work, we refine the circuit-based technique of Beimel and Weinreb [BW06] for building weighted threshold secret sharing schemes, obtaining substantial improvements in two directions: computational schemes that are feasible in practice and information-theoretic schemes for ramp weighted access structures that improve the current constructions.

First, we present an improvement of the computational weighted threshold secret sharing scheme proposed in [BW06] that outperforms the state-of-the-art solutions. In particular, our proposal eliminates the need for shallow monotone sorting networks and gives a precise upper bound on the size of the scheme's public information, lowering the existing upper bounds. Our main result is as follows.

**Theorem 1.1 (Informal).** *Let $\lambda \in \mathbb{N}$ be the security parameter. Under the assumption that one-way functions exist, any weighted threshold access structure with threshold $\sigma$ over $n$ parties admits a computational secret sharing scheme in which the size of each share is $\lambda$ and the size of the public information is $O(\lambda n^2 \log \sigma)$. Moreover, any authorized subset only needs to download $O(\lambda n \log \sigma)$ bits of the public information to recover the secret.*

Since any WTAS admits an equivalent representation in which the total weight is at most $W = 2^{\lceil n \log n \rceil}$ [Mur71], in the worst case the computational scheme of Theorem 1.1 achieves share size $\lambda$, public information size $\tilde{O}(\lambda n^3)$, and requires any authorized subset to download $\tilde{O}(\lambda n^2)$ bits of public information for reconstruction. This result and previous best solutions are presented in Fig. 1.

Moreover, we show that for practical weight distributions, namely those arising in the most widely used blockchain networks [Yak17,Lab17,RYS+20,Lab22], our scheme achieves better parameters than the state-of-the-art solutions. In particular, the total share size in our scheme is smaller than that of the best existing construction by a factor from 300× to 6700×, depending on the specific weight distribution. This comparison is illustrated in Fig. 5.

To obtain the scheme of Theorem 1.1, we use a general version of Yao's compiler [Yao89] that supports threshold gates [Bei25], which reduces the problem of constructing secret sharing schemes for WTASs to the construction of small monotone Boolean circuits realizing monotone weighted threshold functions. This problem has been thoroughly studied in the past, giving positive and negative results [BW06,COS17].

In this context, our main technical contribution is the description of a new monotone Boolean circuit that improves and simplifies the one given by Beimel and Weinreb [BW06] by using threshold gates instead of AND and OR gates. With that, we can avoid the use of shallow monotone sorting networks and give a precise upper bound on the number of gates and wires of the circuit. Our main technical result is the following.

**Theorem 1.2 (Informal).** *Any monotone weighted threshold function with $n$ variables and threshold $\sigma$ can be computed by a monotone Boolean circuit with $O(n \log \sigma)$ gates and $O(n^2 \log \sigma)$ wires.*

| Work | Share Size | Public Information Size | Downloaded Public Information Size | Privacy |
|---|---|---|---|---|
| [BW06] | $\lambda$ | $O(\lambda(n^2+s(n))\log\sigma) = O(\lambda n(n^2+s(n))\log n)$ | | Computational (OWF) |
| [ABI+23,FG24] | | $O(\lambda(n^2+g(n))\log\sigma) = O(\lambda n(n^2+g(n))\log n)$ | | Computational (RSA) |
| Theorem 1.1 | | $O(\lambda n^2\log\sigma) = O(\lambda n^3\log n)$ | $O(\lambda n\log\sigma) = O(\lambda n^2\log n)$ | Computational (OWF) |
| [Sha79] | $w\log W = 2^{O(n\log n)}$ | - | - | Information-Theoretic (Perfect) |
| [BW06] | $n^{O(\log\log\sigma)} = n^{O(\log n)}$ | | | |

Fig. 1: Summary of best weighted threshold schemes. The second, third, and fourth columns provide upper bounds on the share size, public information size, and downloaded public information size, respectively. These bounds assume a 1-bit secret and use the following notation: $\lambda$ for the security parameter, $n$ for the number of parties, $w$ for the weight of a party, $W$ for the sum of the weights of all $n$ parties, and $\sigma$ for the threshold. To avoid inaccuracies caused by the intricate description of monotone sorting networks, we do not give an exact upper bound on the public information size and downloaded public information size for the first two proposals. Instead, we use $s(n)$ and $g(n)$ to denote the number of wires and gates in these networks. Furthermore, we use the fact that the best upper bound for the weights and the threshold in WTASs is $w, \sigma \leq 2^{\lceil n\log n\rceil}$ [Mur71], which are tight [Hås94]. The fourth column specifies the size of the public information needed during the reconstruction process. The *Privacy* column specifies whether the scheme achieves perfect or computational security; in the latter case, the underlying cryptographic assumption is also noted.

The practicality and relevance of this circuit are further reinforced when combined with the recent constructions of Goel, Sasmal, and Wang [GSW26]. In particular, it directly improves their computational scheme for WTASs with homomorphic properties, which relies on the monotone circuit of Beimel and Weinreb [BW06] as a core component. Replacing that component with our circuit yields a more efficient and practical instantiation of their scheme, which allows to build a weighted multiparty computation protocol, and weighted threshold encryption and signature schemes. Further details are provided in Section 1.4.

Next, building on this monotone Boolean circuit and combining it with distinct techniques, we obtain further optimizations that lead to more efficient information-theoretic secret sharing schemes for WTASs.

In this regard, our main contribution is on the construction of efficient schemes for ramp WTASs. We present a novel method for finding weighted threshold schemes with small weights that satisfy the privacy and correctness restrictions posed by the ramp WTAS, even if the gap is small. Specifically, we adapt our majority circuit construction to achieve a scheme with almost linear share size. The statement follows.

**Theorem 1.3 (Informal).** *Let $\sigma$ be a threshold and let $\epsilon \in (0,1]$. Any $(\sigma, (1+\epsilon)\sigma)$-ramp weighted threshold access structure over $n$ parties admits an information-theoretic secret sharing scheme with share size $\tilde{O}(n/\epsilon)$.*

A comparison of our result with the state-of-the-art solutions for ramp WTASs is presented in Fig. 2.

| **Work** | **Share Size** | **Thresholds Gap** | **Privacy** |
|---|---|---|---|
| [GJM+23] | $w = 2^{O(n \log n)}$ | $(t, t + \Omega(\lambda))$ | Statistical |
| [BHS23] (Rounding), [TF24] | $\tilde{O}\left(\frac{n}{\beta-\alpha}\right)$ | $(\alpha W, \beta W)$ | Perfect |
| [BHS23] (BS Channels) | $\max\left\{\lambda^2, \text{poly}\left(\frac{1}{\beta-\alpha}\right)\right\}$ | $(\alpha W, \beta W)$ | Statistical |
| Theorem 1.3 | $\tilde{O}\left(\frac{n}{\epsilon}\right)$ | $(\sigma, (1+\epsilon)\sigma)$ | Perfect |

Fig. 2: Summary of best information-theoretic schemes for ramp WTAS. In the second column, we give an upper bound on the share size of the schemes, considering secrets of 1-bit. Again, we use the fact that the best upper bound for the size of the weights in WTASs is $w = 2^{O(n \log n)}$ [Mur71], which is tight [Hås94]. In the *Thresholds Gap* column, we specify the privacy and reconstruction thresholds of the ramp WTAS. In the second and third rows, it holds that $0 \leq \alpha < \beta \leq 1$. In the *Privacy* column, we distinguish between perfect and statistical security.

Finally, by transforming our monotone majority Boolean circuit into a monotone Boolean formula we enable the construction of an information-theoretically secure scheme for WTASs with share size $n^{O(\log \log \sigma)}$. This matches the best-known construction from [BW06] and is presented in detail in Section C. As in the computational setting, our information-theoretic construction avoids the use of shallow monotone sorting networks, which simplifies the overall design and improves its practicality.

We conclude by noting that, although our schemes target single-bit secrets, they extend naturally to larger ones. The computational scheme supports $\lambda$-bit secrets, while the information-theoretic scheme supports $O(\log n)$-bit secrets. Moreover, both can handle arbitrary-length secrets by applying the scheme independently bitwise.

### 1.2 Open Questions

Our results leave open several questions about the construction of efficient schemes for WTASs. In the following, we highlight four areas where future work could lead to significant improvements of state-of-the-art solutions.

**Computational Weighted Threshold Schemes** The amount of public information required remains quadratic in the number of parties in the worst case. Reducing this overhead to linear or polylogarithmic would greatly enhance the efficiency and usability of such schemes. In this context, adapting the construction of Applebaum et al. [ABI+23] to threshold gates could lead to a linear

reduction in the size of the public information. Moreover, in practical settings, leveraging prior knowledge about the distribution of weights may lead to the design of even more efficient schemes tailored to specific scenarios.

**Information-Theoretic Weighted Threshold Schemes** The best-known schemes for WTASs still have quasipolynomial share size in the number of parties. Moreover, a significant gap remains between this upper bound and the best-known lower bound of $\Omega(\sqrt{n})$ [PS00], even when considering only linear schemes. Closing this gap is a major open challenge: improving the upper bound would enhance the practical viability of these schemes, while tightening the lower bound would provide deeper insights into the inherent limitations of these constructions.

**Complexity Theory** One of the main open questions regarding monotone weighted threshold functions is whether they belong to the class $m\mathcal{NC}^1$. Beimel and Weinreb [BW06] showed that these functions are in $m\mathcal{AC}^1$, but both their circuit and ours require a non-constant fan-in. Notably, constructing an $m\mathcal{NC}^1$ circuit that realizes any weighted threshold function would lead to information-theoretic secret sharing schemes for WTASs with polynomial share size.

### 1.3 Our Techniques

In order to construct our weighted threshold schemes, we follow a two-step procedure. First, we view the access structure as a monotone Boolean function and construct efficient representations of it in terms of monotone majority Boolean circuits, that is, monotone Boolean circuits that also use threshold gates. Later, we apply known cryptographic techniques to convert these alternative representations of our access structure into secret sharing schemes. Next, we detail these steps.

**Monotone Majority Boolean Circuits for Weighted Threshold Functions** Our approach begins with a reduction from any weighted threshold function to a canonical form, where the weights follow a structured pattern of decreasing powers of two. This transformation follows a similar procedure to that of Beimel and Weinreb [BW06]. First, each original weight is decomposed into a sum of distinct powers of two. Then, a new variable is assigned to each of these summands. Finally, additional variables are introduced to ensure that the threshold itself becomes a power of two. This reduction simplifies the subsequent circuit construction by structuring the weights in a way that naturally aligns with binary computations.

With this reduction in place, we construct a polynomial-size monotone majority Boolean circuit that efficiently computes the transformed weighted threshold function. This circuit is designed to perform binary addition using only threshold gates, making it a particularly efficient construction. More specifically, at

each level, threshold gates act as counters, summing input variables of specific weights. Their outputs are then fed into the threshold gates of the next level, propagating carry values upward through successive counting stages. Ultimately, if the cumulative weighted sum of the inputs exceeds the threshold, the final 2-threshold gate at the top level outputs a one.

Our circuit can be viewed as a refined and simplified variant of the one proposed by Beimel and Weinreb [BW06], which also computes weighted threshold functions but relies solely on AND and OR gates rather than threshold gates. Our modifications eliminate the need for the sorting networks required in their construction while providing a precise upper bound on the number of gates and wires in the circuit.

Moreover, we further modify the circuit to reduce by a linear factor the public information needed for each authorized subset. In particular, by adding a small number of OR gates and properly wiring the circuit, we can ensure that the reconstruction process only relies on the wires of a single gate per level, significantly reducing the amount of public information needed to recover the secret.

**Secret Sharing Schemes Construction** Once we have derived a polynomial-size monotone majority Boolean circuit for WTASs, we construct computational weighted threshold schemes and the information-theoretic ramp weighted threshold schemes. In both cases, we apply already known compilers.

In the computational setting, we construct our scheme in Theorem 1.1 by applying a generalized version of Yao's compiler [Yao89,Bei25], which transforms monotone Boolean circuits into computational secret sharing schemes. This version has been extended to support threshold gates and is applied directly to our original monotone majority Boolean circuit.

Regarding the ramp weighted threshold scheme, our starting point is the canonical majority circuit determined by weights $\boldsymbol{w}$ and threshold $\sigma$. Given a parameter $\epsilon$, we apply an appropriate horizontal cut to the circuit, deleting the part of the circuit corresponding to lowest weights. With that, we obtain the circuit of a WTF $f'$ with smaller weights $\boldsymbol{w}'$ and smaller threshold $\sigma'$. For every $\boldsymbol{x} \in \{0,1\}^n$, it satisfies the following properties:

1. If $\boldsymbol{w} \cdot \boldsymbol{x} < \sigma$, then $f'(\boldsymbol{x}) = 0$.
2. If $\boldsymbol{w} \cdot \boldsymbol{x} \geq (1+\epsilon)\sigma$, then $f'(\boldsymbol{x}) = 1$.

Therefore, $f'$ satisfies the $(\sigma, (1+\epsilon)\sigma)$-ramp WTAS. The interest of this approach lies in its alternative weight-rounding method, which produces significantly smaller weights $\boldsymbol{w}'$. Finally, it suffices to apply Shamir's scheme [Sha79] to this access structure to get the construction in Theorem 1.3.

For the information-theoretic scheme for (non-ramp) WTASs, we first apply the same balancing and dynamic programming techniques as Beimel and Weinreb [BW06] to our monotone majority Boolean circuit. This allows us to reduce its depth to logarithmic while preserving its polynomial size, enabling

a black-box transformation into a monotone Boolean formula of quasipolynomial size. From there, we apply the technique of Benaloh and Leichter [BL88] for constructing secret sharing schemes from monotone Boolean formulas, where the share size corresponds to the length of the formula.

### 1.4 Related Work

In this section, we focus on prior work related to the construction of monotone Boolean circuits for monotone weighted threshold functions and the development of weighted threshold schemes.

**Monotone Boolean Circuits for Monotone Weighted Threshold Functions** The study of monotone Boolean circuits consists of understanding the complexity of computing monotone Boolean functions, i.e., Boolean functions that can be expressed using only AND and OR gates. Early foundational work in this area by Razborov [Raz87] and by Alon and Boppana [AB87] established crucial lower bounds for monotone Boolean circuits, revealing that certain monotone Boolean functions, such as clique, require monotone Boolean circuits of exponential size. These results highlighted the limitations of purely monotone constructions for certain classes of Boolean functions.

Goldmann and Karpinski [GK93] showed that weighted threshold gates can be simulated by polynomial-size monotone majority circuits, but their constructions are inherently non-monotone, introducing negations even for monotone functions. This raised the question of whether efficient monotone simulations of monotone weighted threshold functions were possible, particularly in constant depth. Hofmeister [Hof92] provided some progress by demonstrating that monotone, depth-2 Boolean circuits for specific monotone weighted threshold functions would require exponential size, emphasizing the limitations of low-depth monotone Boolean circuits.

In an independent line of work, Beimel and Weinreb [BW06] constructed polynomial-size logarithmic-depth monotone Boolean circuits for any weighted threshold function. Recently, this construction has been proved to be asymptotically optimal, since Chen, Oliveira and Servedio [COS17] answered negatively the question posed by Goldmann and Karpinski [GK93] by proving that there exists a monotone weighted threshold function such that any constant-depth monotone circuit realizing it requires exponential size. Furthermore, they also provide a description of a polynomial-size logarithmic-depth monotone majority Boolean circuit realizing monotone weighted threshold functions.

**Weighted Threshold Secret Sharing Schemes** In 1979, Shamir [Sha79] and Blakley [Bla79] introduced the first secret sharing schemes for threshold access structures. Shamir also proposed a method for realizing WTASs using threshold schemes through virtualization: given weights $w_i$ and a threshold $\sigma$, the dealer treats party $i$ as $w_i$ distinct parties, distributing $w_i$ separate shares of the $\sigma$-threshold scheme to that party. This approach results in a share size of

$O(w_i \log W)$ for any WTAS, where $W$ is the sum of the weights of the parties. However, in the worst-case scenario, that is, when the weights are exponential in the number of parties [Hås94], this scheme produces larger shares than those of the best known construction for general access structures[AN21].

Beimel and Weinreb [BW06] presented constructions based on a polynomial-size logarithmic-depth monotone Boolean circuit that realizes these access structures. By applying Yao's technique [Yao89] for monotone circuits, they obtained a scheme that is computationally secure. Specifically, assuming the existence of secure one-way functions, their scheme achieves shares of linear size in the security parameter and public information of polynomial size in the number of parties. Moreover, the recent work of Farràs and Guiot [FG24] also presents a computational scheme for WTASs that achieves polylogarithmic share size and public information of polynomial size. This scheme is a direct instantiation of the general construction of Applebaum et al. [ABI+23] to the weighted threshold case.

However, these constructions require the use of shallow monotone sorting networks, which are considered impractical due to their intricate description and the large constants (~1830 for sufficiently many inputs) hidden in the big $O$ notation of their circuit complexity [AKS83,Pat90,Chv92]. The complexity of these networks not only complicates the actual implementation of the scheme, but also results in a significant overhead in the size of the public information, as the large hidden constants are directly carried over into the scheme.

By converting this circuit into a monotone Boolean formula and applying the compiler of Benaloh and Leichter [BL88], Beimel and Weinreb [BW06] provided an information-theoretic scheme for WTASs. Implicitly, they show that any WTAS with threshold $\sigma$ admits a scheme with share size $n^{O(\log \log \sigma)}$. Since the best upper bound for the threshold is $\sigma = 2^{\lceil n \log n \rceil}$, their result implies that every WTAS admits a perfect scheme of share size $n^{O(\log n)}$. These are the best information-theoretic and computational schemes for WTASs known to date.

This information-theoretic scheme highlights that WTASs occupy a *privileged* position in terms of efficiency, as most general access structures require linear schemes with a normalized share size of $2^{n/3+o(n)}$ [BF20]. The best known lower bound on the information ratio for these access structures is $\Omega(\sqrt{n})$, established by Padró and Sáez [PS00] through an analysis of WTASs with only two distinct weights. This also remains as the best lower bound on share size to date. Characterizations of ideal WTASs are provided in [BTW08,FP12]. For a broader overview of previous results on weighted and hierarchical access structures, see [FG24].

Recently, Goel, Sasmal, and Wang [GSW26] presented a new computational scheme for weighted threshold access structures based on the Learning With Errors and the Decisional Composite Residuosity assumptions. Their approach follows the paradigm of Yao's compiler [Yao89] for monotone Boolean circuits, augmenting it with a special form of linear homomorphism to obtain a construction that is particularly well-suited for the design of secure multiparty computation protocols. In their instantiation, the underlying representation of WTASs is

given by the monotone Boolean circuit of Beimel and Weinreb [BW06]. In this context, our results are complementary to [GSW26], as we provide an alternative monotone Boolean circuit for WTASs with improved complexity. In particular, our construction can be used as a drop-in replacement for the [BW06] circuit in their framework, leading to a more efficient and practical realization of their secret sharing scheme, their weighted multiparty computation protocol, and their weighted threshold encryption and signature schemes.

**Ramp Weighted Threshold Secret Sharing Schemes** Recently, Benhamouda, Halevi, and Stambler [BHS23], Garg et al. [GJM+23], and Tonkikh and Freitas [TF24] explored a relaxed weighted threshold model, considering ramp WTASs. Since this setting is less restrictive, these works obtain schemes with considerably smaller share size than those for WTASs.

The schemes of Benhamouda, Halevi, and Stambler [BHS23] consider ramp WTASs with privacy threshold $\alpha W$ and reconstruction threshold $\beta W$, for parameters $0 \leq \alpha < \beta \leq 1$. Their first construction relies on rounding the party weights, which yields a scheme with total share size $\tilde{O}\left(\frac{n}{\beta-\alpha}\right)$. Notice that they cannot provide a better bound for the size of each share. Their second construction establishes a novel connection between wiretap channels and secret sharing, reducing the size of each share. By choosing appropriate parameters, this approach achieves $2^{-\lambda}$-statistical security and share size $\max\left\{\lambda^2, \text{poly}\left(\frac{1}{\beta-\alpha}\right)\right\}$.

The construction of Garg et al. [GJM+23] is based on a primitive whose security follows from the Chinese Remainder Theorem [Mig83]. Their scheme provides statistical privacy, and the gap between the privacy and reconstruction thresholds depends on the security parameter $\lambda$. Under this relaxed security notion, they improve upon the share-size bound of Shamir's scheme by a logarithmic factor, obtaining a share size of $O(w)$. Although the resulting scheme is not linear, it can still be used as a building block in more complex protocols.

Finally, Tonkikh and Freitas [TF24] propose a general transformation that maps large real weights to smaller integer weights in a broad class of weighted distributed protocols. In the context of secret sharing, their method produces a ramp WTAS with small weights while preserving the original authorized and forbidden subsets. The resulting share size and thresholds gap match those of the first construction of Benhamouda, Halevi, and Stambler [BHS23].

### 1.5 Organization

In Section 2, we lay out the preliminaries on the complexity of Boolean functions and secret sharing schemes. In Section 3, we construct the polynomial-size monotone majority Boolean circuit of Theorem 1.2 and we prove Theorem 1.1. In Section 4, we compare our computational scheme with the state-of-the-art constructions, analyze the resulting share sizes for weight distributions arising in the most widely used blockchain networks, and provide guidelines for implementing our construction. In Section 5, we present our ramp weighted threshold

secret sharing scheme, proving Theorem 1.3. In Section 6, we conclude the paper and discuss the practicality of our results. The Appendix contains an outline of the general version of Yao's compiler that supports threshold gates (Section A), illustrative examples of our circuit (Section B), the information-theoretic scheme that results from transforming our circuit into a monotone formula (Section C), deferred proofs (Section D), and detailed parameters for the evaluation (Section E).

## 2 Preliminaries

### 2.1 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 $[n]$, and the set $\{0, 1, \ldots, n\}$ as $[n]_0$. For a set $S$, we denote its cardinal as $|S|$. We denote vectors $\boldsymbol{x}$ using bold symbols, their $i$-th coordinates as $x_i$, and their support as $\text{supp}(\boldsymbol{x}) = \{i \in [n] \ : \ x_i \neq 0\}$. 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 their concatenation as $\boldsymbol{x}||\boldsymbol{y}$, and we say that $\boldsymbol{x} \leq \boldsymbol{y}$ if and only if $x_i \leq y_i$ for all $i \in [n]$. For a vector $\boldsymbol{x} \in \mathbb{R}^n$ with indices in the set $[n]$, for every $A \subseteq [n]$ we define $\boldsymbol{x}_A$ as the vector whose elements are the $i$-th coordinates of $\boldsymbol{x}$ with $i \in A$.

Next, we define three functions. We set sign: $\mathbb{R} \to \{0, 1\}$ as the function with $\text{sign}(x) = 1$ if and only if $x \geq 0$; for $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{w} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$, we set $\text{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} - \sigma)$; and for $\boldsymbol{x} \in \{0,1\}^n$ and $\sigma \in \mathbb{R}$, we set $\text{Th}(\sigma)(\boldsymbol{x})$ as the $\sigma$-threshold function.

Finally, we use the $\tilde{O}$ notation to ignore polylogarithmic factors.

### 2.2 Complexity of Boolean Functions

In this section, we present the necessary definitions and technical results on the complexity of Boolean functions. Specifically, we focus on the notion of weighted threshold functions and their properties.

Before proceeding, we formally define monotone majority Boolean circuits.

**Definition 2.1.** *A monotone majority Boolean circuit is a monotone Boolean circuit composed of (unweighted) threshold gates.*

*Remark 2.2.* Note that monotone majority Boolean circuits also include AND and OR gates as special cases: an AND gate is a threshold gate with threshold equal to the fan-in, and an OR gate is obtained by setting the threshold to 1.

We start by defining the main block of our construction.

**Definition 2.3 (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 $f(\boldsymbol{x}) = \text{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$. The vector $(\boldsymbol{w}, \sigma) \in \mathbb{R}^{n+1}$ is said to represent $f$, while the vector $\boldsymbol{x} \in \{0,1\}^n$ is said to be* authorized *(resp.* forbidden*) if $f(\boldsymbol{x}) = 1$ (resp. $f(\boldsymbol{x}) = 0$).*

Our focus is on monotone WTFs, as they are in one-to-one correspondence with WTASs. In this regard, the following remark provides a characterization of monotone WTFs.

*Remark 2.4.* 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 WTFs is that they can be computed by polynomial-size logarithmic-depth monotone circuits of unbounded fan-in [BW06]. Furthermore, it is also known that not all monotone WTFs can be computed by polynomial-size constant-depth monotone circuits of unbounded fan-in [COS17].

**Theorem 2.5 ([BW06,COS17]).** *The family of weighted threshold functions is contained in $m\mathcal{AC}^1$ but it is not in $m\mathcal{AC}^0$.*

Next, we state two useful results about the weights of monotone WTFs. The first is an upper bound on the size of the weights of any monotone WTF. The second one shows that this upper bound is indeed tight.

**Theorem 2.6 ([Mur71]).** *For any monotone WTF $f$ with $n$ variables, there exist $w_1, \ldots, w_n, \sigma \in \mathbb{N}$ smaller than $2^{\lceil n \log n \rceil}$ such that $f(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{w}, \sigma)(\boldsymbol{x})$, where $\boldsymbol{w} = (w_1, \ldots, w_n)$.*

**Theorem 2.7 ([Hås94]).** *For any $n \in \mathbb{N}$ there exists a monotone WTF with $n$ variables requiring integer weights of size $2^{\Omega(n \log n)}$.*

In light of these results, to simplify the notation and without loss of generality, from now on we assume that any WTF has decreasingly ordered integer weights.

We conclude this section by briefly reviewing shallow monotone sorting networks.

**Definition 2.8.** *A sorting network is a Boolean circuit that sorts $n$ input values. It is monotone if it uses only monotone gates (e.g., AND and OR), and shallow if its depth is at most $O(\log n)$.*

Despite extensive research, the best known shallow monotone sorting networks have large constant factors in their depth, limiting their practicality. The current best result is stated next.

**Theorem 2.9.** *[Chv92] For every $n \geq 2^{78}$, there exists a monotone sorting network on $n$ wires with depth at most $1830 \log n - 58657$.*

### 2.3 Secret Sharing Schemes

In the following, we present the basics of 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 2.10 (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*, otherwise we say that $A$ is* forbidden*.*

Definition 2.10 implies that non-constant monotone WTF are a particular case of access structures. In this context, they are also called *weighted threshold access structures* (WTASs).

*Remark 2.11.* In the case of a WTAS given by $f(\boldsymbol{x}) = WTF(\boldsymbol{w}, \sigma)(\boldsymbol{x})$, the family of authorized subsets is

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

This work is dedicated to the following question: Given a WTAS, find an efficient scheme that realizes it. Moreover we also consider a relaxation of this problem: Given a sequence of weights, a privacy threshold $\sigma$ and a reconstruction threshold $\sigma'$, find efficient schemes satisfying that subsets with weight smaller than $\sigma$ are forbidden, while subsets of weight greater or equal than $\sigma'$ are authorized. In this case, we look for an efficient scheme whose access structure satisfies the promise problem determined by the two thresholds and the weights. In this case, by an abuse of notation, we say that the scheme realizes the $(\sigma, \sigma')$-ramp weighted threshold access structure.

**Definition 2.12 (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 a deterministic algorithm* Reconstruct *such that*

- ***Perfect correctness.*** *For any secret $s \in \mathcal{S}$ and any authorized set $A$, it holds that*
$$\mathbf{P}[s = \mathsf{Reconstruct}(\mathsf{Share}(s)_A)] = 1,$$
*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*
$$\mathbf{P}[\{s_i\}_{i \in B} = \mathsf{Share}(s)_B] = \mathbf{P}[\{s_i\}_{i \in B} = \mathsf{Share}(s')_B].$$

**Definition 2.13 (Computational Secret Sharing Scheme,[ABI+23]).** *Let $\mathcal{S} = \{0,1\}$, 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 a deterministic polynomial-time algorithm* Reconstruct *such that*

- ***Correctness.*** *For any secret $s \in \mathcal{S}$, any authorized set $A$, and a description $D$ of $f$ it holds that*

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

  *where $\mathsf{Share}(s, D, \mathbf{1}^\lambda)_A$ denotes the restriction of the output of $\mathsf{Share}(s, D, \mathbf{1}^\lambda)$ to the parties in $A$.*
- ***Privacy.*** *The scheme is $t(\lambda)$-secure if any $t(\lambda)$-time adversary wins the security game of Definition 2.14 with probability at most $\frac{1}{t(\lambda)}$.*

The security of computational secret sharing schemes is given in terms of a game between an adversary and a challenger. The definition follows.

**Definition 2.14 (Security,[ABI+23]).** *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* ($\mathsf{Share}$, $\mathsf{Reconstruct}$) *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 $\mathsf{Share}(s, D, \mathbf{1}^\lambda)$ and sends $\mathsf{Share}(s, D, \mathbf{1}^\lambda)_A$ 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 $\lambda$, 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 $t > \lambda$.*

Next, we introduce the notion of public information, which is crucial when studying the share size of computational secret sharing schemes.

*Remark 2.15.* In computational secret sharing, share sizes can be reduced by using public information accessible to all parties. Instead of privately distributing each share, the dealer encrypts it with a semantically secure symmetric scheme and publishes the encrypted shares as public information, while each party receives only its decryption key. Consequently, efficiency must account for both the share and public information sizes.

We end this section by stating the general version of Yao's compiler [Yao89,Bei25], which constructs computational secret sharing schemes from monotone majority Boolean circuits.

**Theorem 2.16 ([Yao89]).** *Let $\lambda \in \mathbb{N}$ be the security parameter. Under the assumption that one-way functions exist, any access structure over $n$ parties described by a monotone majority Boolean circuit $C$ admits a computational secret sharing scheme with share size $\lambda$ and public information size $O(|C|)$, where $|C|$ denotes the number of wires in $C$.*

*Remark 2.17.* The above theorem generalizes Yao's original result [Yao89], which was stated specifically for monotone Boolean circuits composed solely of AND and OR gates. A formal proof of this general version is provided in [Bei25], while an overview of the underlying idea is presented in Section A.

# 3 Computational Weighted Threshold Secret Sharing Schemes

In this section, we present an improved version of the computational weighted threshold secret sharing scheme proposed by Beimel and Weinreb [BW06]. The main result is the following.

**Theorem 3.1 (Theorem 1.1 restated).** *Let $\lambda \in \mathbb{N}$ be the security parameter and assume that $\lambda = \Omega(\log n)$. Under the assumption that one-way functions exist, any weighted threshold access structure over $n$ parties with threshold $\sigma$ admits a computational secret sharing scheme where the size of the shares is $\lambda$ and the size of the public information is $4\lambda n^2(\log \sigma + 1)$. Moreover, any authorized subset only needs to download at most $4\lambda n(\log \sigma + 1)$ bits of the public information to recover the secret.*

We divide the proof of Theorem 3.1 into three steps. First, we present a reduction from any WTF to a canonical WTF in which the weights are structured as decreasing powers of two. Next, we construct a polynomial-size monotone majority Boolean circuit for this canonical WTF and use it to derive our computational scheme. Finally, we show how to further reduce the public information needed for the secret reconstruction.

## 3.1 Reducing Monotone Weighted Threshold Functions to a Canonical Form

In order to construct a monotone majority Boolean circuit for any monotone WTFs, we first present a reduction from any WTF to a canonical WTF in which the weights are structured as decreasing powers of two. This reduction simplifies the subsequent construction of the circuit by aligning the weights with a structure naturally suited for binary computations.

To do so, we follow an idea similar to the one in [BW06], where they perform a reduction from any monotone WTF to a so-called *universal* monotone WTF. In more detail, given any monotone WTF $f$ over $n$ variables, we convert it into a monotone WTF $g$ over $m = O(n \log \sigma)$ variables such that its threshold and all its weights are decreasing powers of two, and any authorized input $\boldsymbol{x} \in \{0,1\}^n$ for $f$ can be converted to an authorized input $\boldsymbol{y} \in \{0,1\}^m$ for $g$. The following theorem formalizes this reduction and its proof is deferred to Section D.1.

**Theorem 3.2.** *Let $f$ be a monotone WTF over $n$ variables with threshold $\sigma$ and weights $\boldsymbol{w}$. Then there exists $\sigma' < 2\sigma$, a monotone decreasing sequence of $m = O(n \log \sigma)$ powers of two $\boldsymbol{v}$, and an injective mapping $\rho : \{0,1\}^n \to \{0,1\}^m$ for which the monotone WTF $g(\boldsymbol{x}) = \mathrm{WTF}(\boldsymbol{v}, \sigma')(\boldsymbol{x})$ satisfies that $f(\boldsymbol{x}) = g(\rho(\boldsymbol{x}))$.*

Intuitively, the reduction performed in Theorem 3.2 consists of decomposing each original weight as the sum of the various powers of two that add up to it, and then associating a new variable to each of these summands. Finally, we must add some constants set to one (and their associated weights) to ensure that increasing the threshold to the next power of two does not change the structure of the function. By applying this reduction, we transform the problem of computing any monotone WTF into the problem of determining whether the sum of a given set of powers of two exceeds a larger power of two. In practical terms, this reshaping of the weights leads to a structure that is inherently compatible with binary calculations, which facilitates the circuit construction.

### 3.2 Construction of the Scheme

After showing in Theorem 3.2 how to reduce the computation of any monotone WTF to the computation of a monotone WTF in which all the weights are powers of two, we move to construct a monotone majority Boolean circuit for it.

Before formally constructing the circuit, we give a high-level overview of its underlying idea. Intuitively, since by Theorem 3.2 we can assume that the threshold $\sigma$ and the weights are decreasing powers of two ranging from 1 to $2^{\lceil \log \sigma \rceil}$, the circuit simply performs binary addition and checks whether the resulting sum exceeds the threshold $\sigma$. More concretely, the threshold gates at each level function as counters, determining how many variables with a given weight appear at the input. Then, their output wires are fed into the threshold gates of the level above, which allows the circuit to convert the previous sum into the carry value to be used in the counting process of the next level. This process continues hierarchically until the top level, where a single 2-threshold gate evaluates to 1 whenever the sum of the weights of the input variables is above the threshold.

The general construction is presented in Theorem 3.3 and it is outlined in Fig. 3. See Fig. B.1 for a concrete instantiation of the circuit.

**Theorem 3.3 (Theorem 1.2 restated).** *Any monotone WTF over $n$ variables and threshold $\sigma$ can be computed by a monotone majority Boolean circuit with at most $n(\log \sigma + 1)$ gates and $2n^2(\log \sigma + 1)$ wires.*

*Proof.* Let $f$ be a monotone WTF over $n$ variables and threshold $\sigma$. First, recall that by using the reduction from Theorem 3.2 we simply need to build a monotone majority Boolean circuit for the monotone WTF $g$ consisting of $O(n \log \sigma)$ variables and in which the threshold and the weights are decreasing powers of two. In particular, the threshold is set to $\sigma' = 2^{\lceil \log \sigma \rceil}$ and for each weight $w_i$ there are at most $n$ distinct variables.

We now proceed to describe the circuit and prove its correctness. First, recall from Section 3.1 that the computation of $g$ can be interpreted as a binary addition: it checks whether the sum of the powers of two associated with the input variables reaches or exceeds the threshold, which is itself a power of two.

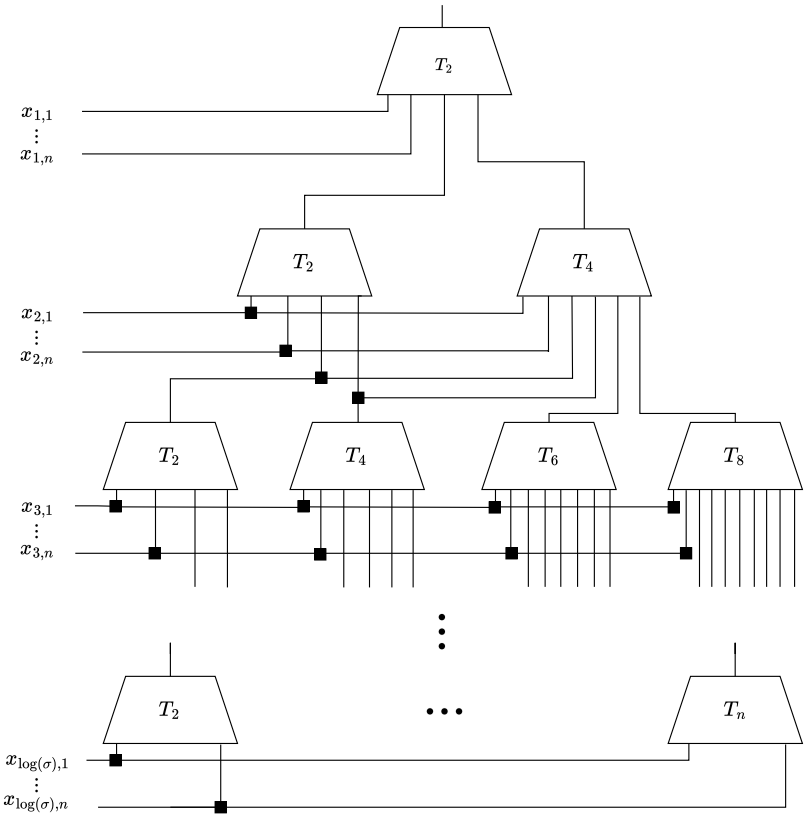


Fig. 3: Monotone majority Boolean circuit realizing a WTAS over $n$ variables.

With this in mind, we organize the variables into levels based on their weights, with the lowest level containing variables of weight one, and the top level containing those of weight $\frac{\sigma'}{2}$. At each level $i$, we introduce a threshold gate for every even value from 2 up to the minimum of $2n$ and $\frac{\sigma'}{w_i}$. The bound of $2n$ ensures that we account for the maximum number of values that can appear at any level (namely, up to $n$ input variables of the same weight plus up to $n$ carry values).[1] The bound $\frac{\sigma'}{w_i}$ reflects the fact that if more than this number of variables with weight $w_i$ are present, the threshold $\sigma'$ is already exceeded, and the circuit can trivially output 1 without further computation.

The purpose of these threshold gates is to count how many variables of a given weight are present in the input and to transfer the resulting value to the next level, simulating the notion of carry bits. To achieve this, the output wires

[1] At the lowest level, there are no carry values, so the upper bound simplifies to $n$.

of the threshold gates at each level are connected as input wires to the threshold gates at the level above, allowing them to act as carry values.[2] The final output of the circuit is given by the output wire of the unique 2-threshold gate at the top level.

To prove its correctness, we follow the same argument as in the work of Beimel and Weinreb [BW06]. For every $i \in [\log \sigma']$, let $a_i$ be the number of variables of level $i$ that appear in the input, let $v_i = \sum_{j \in [i]} 2^{j-1} a_j$, and let $c_i$ be the carry value propagated from the level $i$ to the level $i+1$.[3] We set by definition $c_0 = v_0 = 0$.

First, note that at each level $i$, since there are threshold gates only for each even value up to $\min\{2n, \frac{\sigma'}{w_i}\}$, and the carry values correspond to the outputs of those gates, it holds that $c_i = \left\lfloor \frac{a_i + c_{i-1}}{2} \right\rfloor$.

Furthermore, $c_{\log \sigma'} \in \{0,1\}$ corresponds to the output of the unique 2 threshold gate at the top level, which is the output of the circuit. Therefore, to prove the correctness of the circuit it suffices to check that $c_{\log \sigma'} = 1$ if and only if $v_{\log \sigma'} \geq \sigma'$. To do so, we state and prove a claim that relates the values $v_i$ and $c_i$.

*Claim.* For every $0 \leq i \leq \log \sigma'$ it holds that $v_i - 2^i < 2^i c_i \leq v_i$.

*Proof of the Claim.* We prove the claim by induction. The base case $i = -1$ is correct by inspection. To prove the inductive step, we first notice that by definition it holds that

$$v_i = v_{i-1} + 2^{i-1} a_i.$$

Now, using the induction hypothesis, the first inequality holds because

$$2^i c_i = 2^i \left\lfloor \frac{a_i + c_{i-1}}{2} \right\rfloor \geq 2^i \frac{a_i + c_{i-1} - 1}{2} = \\ = 2^{i-1} a_i + 2^{i-1} c_{i-1} - 2^{i-1} > v_i - 2^i.$$

Finally, using again the induction hypothesis, the second inequality holds because

$$2^i c_i = 2^i \left\lfloor \frac{a_i + c_{i-1}}{2} \right\rfloor \leq 2^i \frac{a_i + c_{i-1}}{2} = \\ = 2^{i-1} a_i + 2^{i-1} c_{i-1} \leq 2^{i-1} a_i + v_{i-1} = v_i.$$

Now, to prove the correctness of the circuit it remains to notice that whenever $v_{\log \sigma'} \geq \sigma'$, the first inequality of the Claim implies that $c_{\log \sigma} = 1$, so the circuit accepts. While when $v_{\log \sigma'} < \sigma'$, the second inequality of the Claim implies that $c_{\log \sigma} = 0$, so the circuit rejects.

[2] More precisely, the output wires of a $t$-threshold gate only need to be connected to those gates at the next level whose thresholds are at least $\frac{t}{2}$. However, since this optimization does not significantly affect the overall circuit complexity, we omit the details for simplicity.

[3] By construction of the circuit, $c_i$ is the number of output wires set to 1 at level $i$.

It only remains to prove the bound on the number of gates and wires of the circuit. From its description, it is clear that the circuit has at most $\log \sigma' \leq \log \sigma + 1 = O(\log \sigma)$ levels. In each level there are at most $n$ distinct input variables and $n$ distinct carry values, which limits the number of gates per level to $n$. Hence, there are at most $n \log \sigma' \leq n(\log \sigma + 1) = O(n \log \sigma)$ gates in the circuit. Moreover each gate has at most $2n$ input wires, which correspond to the $n$ variables and the $n$ carry values per level. Therefore, there are at most $2n^2 \log \sigma' \leq 2n^2(\log \sigma + 1) = O(n^2 \log \sigma)$ wires in the circuit.

*Remark 3.4.* Notice that in the proof of Theorem 3.3, if instead of taking the threshold $\sigma$ as a parameter, we replace it by its upper bound of $\lceil n \log n \rceil$ given by Theorem 2.6, the depth of the circuit can be upper bounded by $O(n \log n)$. This leads to the upper bounds of $O(n^2 \log n)$ and $O(n^3 \log n)$ for the number of gates and wires, respectively.

Once we have constructed a polynomial-size monotone majority Boolean circuit for any WTF, we simply apply the general version of Yao's compiler [Yao89], extended to support threshold gates [Bei25], to construct a computational secret sharing scheme for WTASs. This is summarized in the following theorem and its proof is deferred to Section D.2.

**Proposition 3.5.** *Let $\lambda \in \mathbb{N}$ be the security parameter and assume that $\lambda = \Omega(\log n)$. Under the assumption that one-way functions exist, any weighted threshold access structure over $n$ parties with threshold $\sigma$ admits a computational secret sharing scheme where the size of the shares is $\lambda$ and the size of the public information is at most $2\lambda n^2(\log \sigma + 1)$.*

*Remark 3.6.* As in Remark 3.4, a general upper bound on the public information is $O(\lambda n^3 \log n)$.

### 3.3 Reduction of the Public Information

We now show how to reduce the amount of public information needed for each authorized subset during the reconstruction process by a linear factor in $n$ at the cost of increasing its total size by a constant factor. To do so, we slightly modify the circuit constructed in Section 3.2 by adding to it $O(n \log \sigma)$ additional OR gates and $O(n^2 \log \sigma)$ new wires.

Next, we outline its core idea. The circuit, as constructed in Theorem 3.3, requires the parties to use the wires of all gates whose output is set to 1 to recover the secret. However, this approach is inefficient in terms of the number of gates used. Essentially, at any given level, among all gates whose output is set to 1, only the gate with the highest threshold matters, as it dominates the others. Specifically, if the output of a gate is 1, then all gates with lower thresholds at that level also have their output set to 1.

To optimize this, we ensure that each authorized subset only considers the wires of these critical gates. This is achieved by connecting the output wire of each threshold gate to an OR gate, which also receives as input the output wires

of all threshold gates with higher thresholds from the same level. In this way, during the reconstruction process, at any level it suffices to look at the public information associated to the wires of a single gate rather than that of all $n$ gates.

Fig. 7 corresponds to the optimization of the circuit of Fig. 6 and Fig. B.2 contains a concrete instantiation. The result is stated as follows and its proof is deferred to Section D.3.

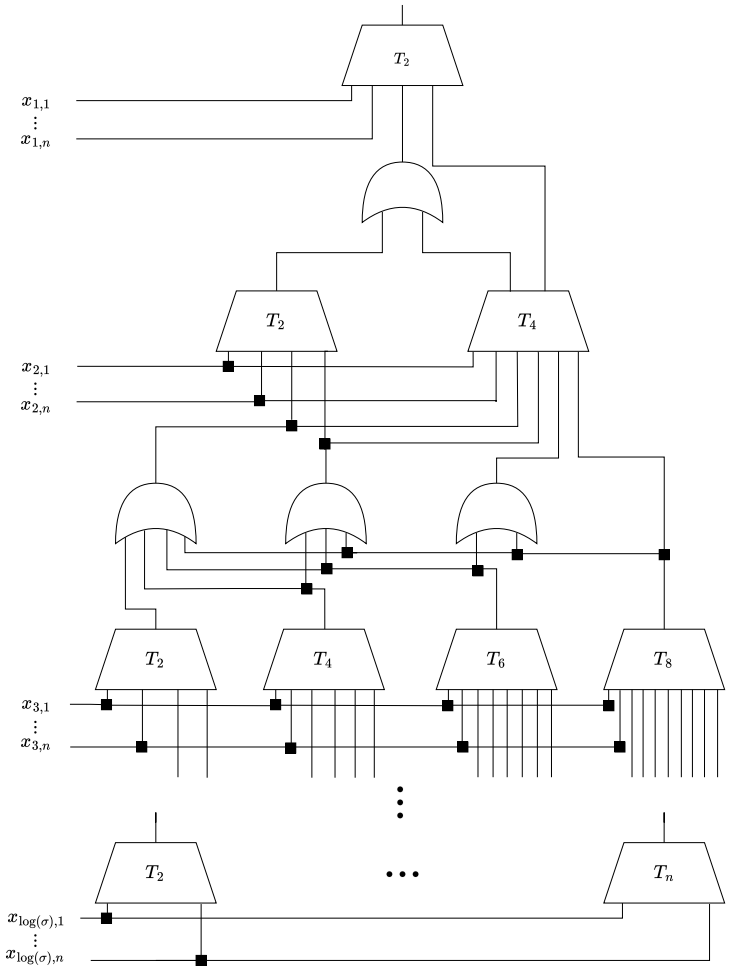


Fig. 4: Optimized version of the circuit realizing a WTAS over $n$ variables.

**Proposition 3.7.** *At the cost of increasing the size of the public information in Theorem 3.5 by a factor of two, the amount of public information that each authorized subset needs to download during the reconstruction process is at most* $4\lambda n(\log \sigma + 1)$. *Moreover, each party only needs to download* $4\lambda(\log \sigma + 1)$ *bits, in average.*

From a theoretical perspective, the circuit constructed in Theorem 3.7 is worse than that in Theorem 3.3, since it uses about twice as many gates and wires to compute the same function. However, from a practical approach, it is more efficient because it allows any authorized subset to recover the secret using less data and computation. In this sense, it is useful to think of public information as a huge dataset that the dealer publishes during the sharing phase over the Internet. For example, the dealer uploads it to a server. With this in mind, the circuit constructed in Theorem 3.7 allows any authorized subset of parties to download, during the reconstruction phase, only the public information corresponding to the wires of a single gate per level, which reduces the data that they must handle by a linear factor. In addition, this also reduces by the same proportion the amount of computations the parties need to perform, since they only need to compute one polynomial interpolation per level.

## 4 Technical Comparison, Evaluation, and Implementation Guidelines

In this section, we compare our scheme with the state-of-the-art constructions. First, we highlight the technical advantages of our approach over the existing computational solutions: the scheme by Beimel and Weinreb [BW06], and the scheme by Farràs and Guiot [FG24], which is a direct instantiation of the general construction of Applebaum et al. [ABI+23] to the weighted threshold case. Later, we empirically analyze and compare the share sizes of all these schemes for weight distributions arising from the most widely used blockchain networks [Yak17,Lab17,RYS+20,Lab22].

### 4.1 Technical Comparison

A comparison of our result with the state-of-the-art solutions for WTASs is presented in Fig. 1.

**Boolean Circuit** The primary distinction between the construction of Theorem 3.1 and the previous computational schemes [BW06,FG24] lies in the design of the underlying monotone Boolean circuits used to compute monotone WTFs. Specifically, their constructions rely on circuits that perform binary addition using only AND and OR gates, combined with shallow monotone sorting networks. However, these sorting networks are intricate to describe and implement [AKS83,Val84], and their complexity hides huge constants (~1830 for large values of $n$) that make them impractical to use in the real world [Pat90,Chv92].

In contrast, our approach eliminates the need for these sorting networks by using threshold gates to perform binary addition directly. This produces a monotone majority Boolean circuit that not only simplifies but also enhances the design proposed by Beimel and Weinreb, making it far more suitable for practical applications.

**Share Size** In all three schemes, the share size is equal to the security parameter, as they all rely on public information. However, our scheme stands out because its public information size does not include any factor depending on the complexity of shallow monotone sorting networks. This not only reduces the public information size but also allows us to provide an explicit and concrete upper bound through a direct count of gates and wires. In particular, as shown in Theorem 3.1, the hidden constant in our big $O$ bound is only 2, compared to the much larger constant (~1830) in the other proposals. In addition, the optimization in Proposition 3.7 further reduces the required public information in our construction by a linear factor, making it even more efficient in practice than existing solutions. As a side note, the public information in [FG24] scales linearly with the number of gates, while both Theorem 3.1 and [BW06] scale linearly with the number of wires, which for some circuits may yield a quadratic improvement [ABI+23].

**Cryptographic Assumptions** Another key difference lies in the cryptographic assumptions. Our proposal and the scheme in [BW06] rely on the existence of secure one-way functions, whereas the construction in [FG24] is based on the stronger RSA assumption. This divergence arises from the use of different secret sharing compilers for monotone Boolean circuits: the first two employ Yao's technique [Yao89], while the latter is built on the results from Applebaum et al. [ABI+23].

### 4.2 Evaluation

We empirically compare all size metrics of our scheme with those of the state-of-the-art solutions, namely the virtualization version of Shamir [Sha79], and the computational schemes of Beimel and Weinreb [BW06], and Farràs and Guiot [FG24], using weight distributions from the most widely used blockchain networks [Yak17,Lab17,RYS+20,Lab22].

For the evaluation, we consider 1-bit secrets, a security level of 128 bits, and measure four metrics: *share size*, *public information size*, *downloaded public information size*, and *total share size*. The share size is the number of bits each party receives from the dealer during the sharing phase. The public information size is the number of bits the dealer makes public during the sharing phase. The downloaded public information is the number of bits each party must retrieve from the public information during reconstruction. Finally, the total share size is the number of bits each party needs during reconstruction, which equals the sum of the share size and the downloaded public information size.

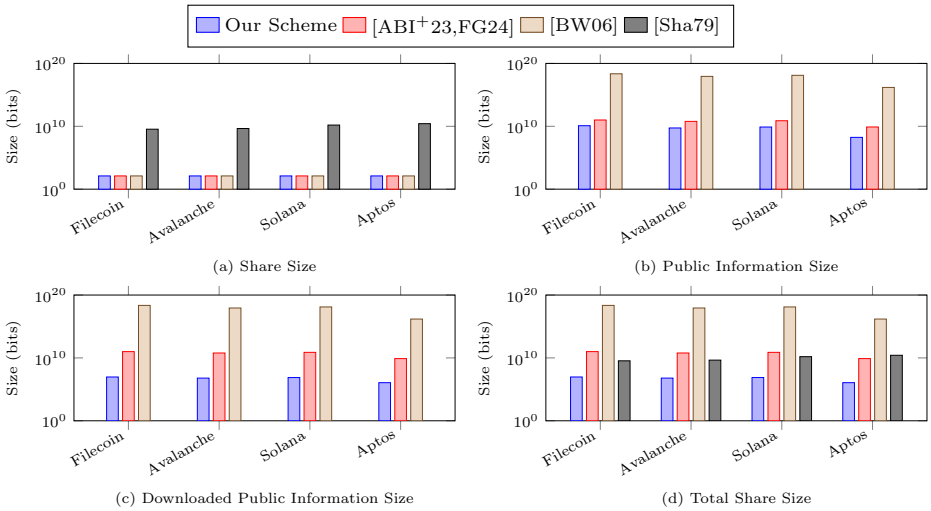


Fig. 5: Comparison of all size metrics between our scheme and the state-of-the-art constructions, using weight distributions from [Fil,Ava,Sol,Apt], considering 1-bit secrets, and a security level of 128 bits. The total share size is defined as the sum of the share size and the downloaded public information size.

We compare our scheme with three state-of-the-art constructions: the virtualization version of Shamir [Sha79], the scheme of Beimel and Weinreb [BW06], and the scheme of Farràs and Guiot [FG24]. The metrics for these proposals are evaluated using the weight distributions of four of the most widely used blockchain networks: Filecoin [Lab17,Fil], Avalanche [RYS+20,Ava], Solana [Yak17,Sol], and Aptos [Lab22,Apt].

We report our results in Fig. 5 to show that our scheme reduces the total share size compared to state-of-the-art solutions. The results are computed exactly for each metric across the different weight distributions. For our scheme, the public information is calculated using the precise expression of Theorem 3.1. For the schemes of Beimel and Weinreb [BW06] and Farràs and Guiot [FG24], we compute a conservative lower bound for the public information size by taking the raw expression of Fig. 1 without constants, while replacing the factors $g(n)$ and $s(n)$ with their exact values ($g(n) = 1830n \log n$ and $s(n) = (1830n \log n)^2$), which is justified since the number of parties in all weight distributions is sufficiently large ($\gg 100$) [AKS83,Pat90,Chv92]. For Shamir's scheme [Sha79], where the size of the shares varies by party according to $w \log W$, we report the value corresponding to the party with the largest weight.

Section E in the appendix provides all the parameters and formulas required to reproduce the results shown in Fig. 5.

**Share Size** As expected, the share size in Shamir's scheme is much larger than in the computational schemes, since the latter rely on public information. In

contrast, all computational schemes have the same share size, which equals the security parameter.

**Public Information Size** Our scheme achieves a much smaller public information size than the other computational schemes. This is because, due to our circuit description, we avoid the large constant factor of 1830 introduced by shallow monotone sorting networks. The notable gap between the public information size of the scheme of Beimel and Weinreb and that of Farràs and Guiot arises from the fact that the former scales linearly with the number of wires, whereas the latter scales linearly with the number of gates, which for these circuits leads to a quadratic improvement. In contrast, Shamir's scheme has no public information at all, as it is purely information-theoretic.

**Downloaded Public Information Size** Thanks to the optimization in Theorem 3.7, our scheme reduces the downloaded public information size by a linear factor in the number of parties compared to the total public information size. In contrast, the other computational schemes do not benefit from this optimization, so their downloaded public information size remains equal to their total public information size. As in the previous case, Shamir's scheme has no public information, since it is information-theoretic.

**Total Share Size** Our scheme achieves a significantly smaller total share size than all other proposals across all weight distributions, clearly demonstrating its efficiency. In particular, it reduces the total share size by factors of 373×, 723×, 2085×, and 6773× (from left to right) compared to the next best construction. Notably, in all distributions except Aptos, Shamir's scheme outperforms the other computational constructions, highlighting their inefficiency due to the large constant factors in their circuit descriptions.

### 4.3 Implementation Guidelines

At a high level, implementing our computational scheme requires three steps: convert the WTF to its canonical power-of-two form (Theorem 3.2), build the monotone circuit (Theorem 3.3), and apply Yao's compiler (Theorem 2.16). We now outline design choices for each step.

**Reduction to the Canonical WTF** It can be implemented with a simple procedure. First, lift the threshold to the next power of two by adding an extra party whose weight equals the difference between the new and original thresholds. Then, perform a binary decomposition of each weight, splitting it into a sum of distinct powers of two and introducing a corresponding variable for each component.

**Circuit Design** The circuit design can be naturally implemented as a directed graph. In this representation, each gate is modeled as a node storing its type (threshold or OR), a list of input gates, and a list of output gates to which it connects. Keeping track of both the input and output connections is important, as the scheme construction requires traversing the circuit in both directions: from top to bottom during the sharing phase and from bottom to top during reconstruction.

**The Scheme** It consists of two procedures: sharing and reconstruction. Sharing proceeds in a top-down manner along the circuit, and computations at each level can be parallelized for efficiency. In particular, it suffices to process the circuit level by level, focusing on the threshold gates. For each such gate, one first generates a $\lambda$-bit secret key for its associated PRG. The PRG and the key are then used to mask the Shamir sharings corresponding to the gates in the level above that are connected to its output wires. After masking, Shamir's sharing is applied over the secret key.

Reconstruction proceeds bottom-up, also allowing parallel computation at each level, and essentially reverses the sharing procedure. At each threshold gate, polynomial interpolation is performed to recover the corresponding $\lambda$-bit key, which is then used with the PRG to unmask the sharings of the gates in the level above for which it serves as input.

For cryptographic soundness, we recommend instantiating the PRG following NIST SP 800-90A guidelines [BK15].

## 5 Schemes for Ramp Weighted Threshold Access Structures

In this section, we adapt our circuit-based construction to obtain a information-theoretic schemes for ramp WTASs with almost linear total share size and small gap. We end this section comparing our proposal with the existing state-of-the-art constructions.

### 5.1 Construction of the Scheme

The circuit from Theorem 3.3 provides an alternative representation of any WTAS. Rather than modifying weights directly, as in prior work [BHS23,TF24], we operate on the circuit, enabling finer control. By truncating the monotone majority circuit to logarithmic depth, we effectively round the weights to near-linear size, with a small loss in resolution, obtaining a scheme for a ramp WTAS.

**Theorem 5.1 (Theorem 1.3 restated).** *Let $\epsilon \in (0, 1]$ be non-negligible in $n$. Any $(\sigma, (1+\epsilon)\sigma)$-ramp weighted threshold access structure over $n$ parties admits an information-theoretic secret sharing scheme with share size $O(\frac{n \log n}{\epsilon})$.*

*Proof.* Let $c \in \mathbb{N}$ be the smallest integer function such that $\frac{1}{c} < \epsilon$, and let $\tau = \lceil \log \sigma \rceil$. Since $\epsilon$ is not negligible, $c$ is polynomial in $n$. We first consider the case that $\sigma$ is a power of 2, and later we show how to deal with the general case.

Consider the monotone majority Boolean circuit for $f$ as defined in Theorem 3.3. Truncating it at depth $\log d = \lceil \log cn \rceil$ yields a circuit representation of a WTF $f'$ with the following properties:

1. $f'$ has the same threshold as $f$, namely $\sigma$.
2. The maximum weight in $f'$ is $\sigma$ and all its weights are multiples of $\sigma/d$.
3. For any $\boldsymbol{x} \in \{0,1\}^n$ with $\boldsymbol{x} \cdot \boldsymbol{w} \geq \sigma + \frac{\sigma}{c}$, it holds that $f'(\boldsymbol{x}) = 1$.

First, we prove properties 1), 2), and 3). Property 1) follows directly from the circuit construction. Property 2) holds because, by design, the maximum weight in $f'$ is $2^\tau$ and all its weights are powers of two. Therefore, each weight is a multiple of the smallest one, which is $2^\tau / 2^{\log d} = \sigma/d$. Property 3) follows from the truncation: the circuit ignores inputs from levels below $\log d$, whose total weight is at most $n\sigma/d \leq \sigma/c$.

Next, observe that conditions 1) and 3), together with the inequality $\sigma + \frac{\sigma}{c} = (1 + \frac{1}{c})\sigma < (1 + \epsilon)\sigma$, imply that any scheme for $f'$ also realizes the $(\sigma, (1 + \epsilon)\sigma)-$ramp WTAS. Hence, it suffices to construct a scheme for $f'$.

By Property 2), we can scale down both the threshold and all the weights of $f'$ by a factor of $\sigma/d$, obtaining an equivalent WTAS whose maximum weight is $d$. Applying Shamir's virtualization technique to this new representation of $f'$ results in a scheme for it with a share size of $d \log dn = O(\frac{n \log n}{\epsilon})$.

We now consider the case where the threshold $\sigma$ is not a power of two. Let $\sigma'$ be the smallest power of two such that $\sigma' \geq \sigma$, and introduce an additional participant with weight $\sigma' - \sigma$. We then apply the same construction to the resulting access structure with threshold $\sigma'$ and this extra participant, obtaining a scheme for the $(\sigma', (1 + \epsilon)\sigma')$-ramp WTAS.

After the sharing phase, we simply make the share of the additional participant public. This effectively removes its contribution, and the resulting scheme realizes the desired $(\sigma, (1 + \epsilon)\sigma)$-ramp WTAS.

### 5.2 Technical Comparison

A comparison between our result and the state-of-the-art constructions for ramp WTASs is shown in Fig. 2. Across all metrics, our scheme consistently matches or improves upon the best existing ramp proposals. We next discuss these comparisons in detail.

**Share Size** For a $(\sigma, (1 + \epsilon)\sigma)$ gap, our scheme attains the smallest share size among existing perfect constructions. In particular, although the scheme of Tonkikh and Freitas [TF24] and the rounding-based construction of Benhamouda, Halevi, and Stambler [BHS23] achieve comparable share sizes, our scheme is strictly more efficient under the same privacy–reconstruction gap.

Specifically, setting $\alpha W = \sigma$ and $\beta W = (1+\epsilon)\sigma$, we obtain $\epsilon = \frac{(\beta-\alpha)W}{\sigma} = \frac{\beta-\alpha}{\alpha}$. Under this parameterization, our scheme achieves a total share size of $O(\frac{\alpha}{\beta-\alpha} n \log \frac{\alpha n}{\beta-\alpha})$, whereas the schemes in [BHS23,TF24] incur a larger share size of $O(\frac{1}{\beta-\alpha} n \log \frac{n}{\beta-\alpha})$. Thus, for the same gap, our construction improves the share size by a factor of $\alpha$, leading to strictly smaller shares. However, unlike their approach, we cannot guarantee a total share size smaller than $O(\frac{n}{\beta-\alpha} \log n)$.

**Thresholds Gap** In terms of the gap between the privacy and reconstruction thresholds, our scheme and the rounding-based schemes of [BHS23] and [TF24] allow arbitrarily small gaps. In contrast, the gap in the construction of Garg et al. [GJM$^+$23] has a lower bound of $\Omega(\lambda)$.

**Privacy** Our construction and the rounding schemes of [BHS23] and [TF24] achieve perfect privacy, while the other two proposals only provide weaker statistical privacy.

*Remark 5.2.* Our circuit-based technique offers a flexible parametrization of the rounding process. With a minor modification, we obtain the rounding schemes of Benhamouda, Halevi, and Stambler [BHS23] and Tonkikh and Freitas [TF24]. In particular, if instead of increasing the threshold $\sigma$ only up to $\sigma'$, we raise it all the way to $W$, we obtain a scheme whose total share size is almost linear in the number of parties, at the cost of a slightly larger individual share size than in the construction of Theorem 1.3. This highlights a natural trade-off between minimizing individual share size and minimizing total share size, and allows the scheme to be tailored to the needs of different applications.

## 6 Conclusion

In this work, we revisit weighted threshold secret sharing from a circuit-based perspective by considering alternative monotone Boolean circuits and formulas that avoid the use of monotone sorting networks. This approach allows us to overcome the main practical limitations of previous constructions, leading to the design of a computational scheme that is feasible in practice, with concrete and efficient parameters. Moreover, by extending these techniques, we obtain an improved information-theoretic ramp weighted threshold secret sharing scheme, achieving the smallest share sizes and threshold gaps among existing solutions.

As shown in Section 4, our computational scheme achieves substantially smaller parameters than state-of-the-art solutions for some of the most widely used blockchain networks, making it a practical alternative for real-world deployments. We also provide guidelines for implementing our computational scheme, and a natural direction for future work is to realize this implementation and deploy it in real-world systems.

The advantages of our proposal are further strengthened when combined with the recent results of Goel, Sasmal, and Wang [GSW26]: by replacing their use

of the [BW06] monotone circuit with our improved construction, one obtains additional efficiency gains in their weighted multiparty computation protocol, as well as in their weighted threshold encryption and signature schemes.

## Acknowledgments

We thank Amos Beimel for valuable comments and suggestions.

## References

AB87. Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 1987.

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

AKS83. M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *STOC*, 1983.

AN21. Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of $1.5^{\mathrm{n}}$. In *CRYPTO*, 2021.

Apt. Aptos stake distribution. `https://aptoscan.com/validators?ps=100&p=`. Accessed: 2025-09-15.

Ava. Avalanche stake distribution. `https://www.avax.network/build/validators`. Accessed: 2025-09-15.

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.

Bei25. Amos Beimel. Secret-sharing schemes for general access structures: An introduction. Cryptology ePrint Archive, Paper 2025/518, 2025.

BF20. Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. Cryptology ePrint Archive, Paper 2020/664, 2020.

BHS23. Fabrice Benhamouda, Shai Halevi, and Lev Stambler. Weighted secret sharing from wiretap channels. In *ITC*, 2023.

BK15. Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators, 2015.

BL88. Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO*, 1988.

Bla79. George Robert Blakley. Safeguarding cryptographic keys. In *AFIPS National Computer Conference*, 1979.

BTW08. Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. *SIAM J. Discrete Math.*, 2008.

BW06. Amos Beimel and E. Weinreb. Monotone circuits for monotone weighted threshold functions. *Inform. Process. Lett.*, 2006.

Chv92. Vasek Chvátal. Lecture notes on the new aks sorting network. 1992.

COS17. Xi Chen, Igor C. Oliveira, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *STOC*, 2017.

FG24. Oriol Farràs and Miquel Guiot. Reducing the share size of weighted threshold secret sharing schemes via chow parameters approximation. In *TCC*, 2024.
Fil. Filecoin stake distribution. `https://filfox.info/en`. Accessed: 2025-09-15.
FP12. Oriol Farràs and Carles Padró. Ideal hierarchical secret sharing schemes. *IEEE Transactions on Information Theory*, 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, *CRYPTO*, 2023.
GK93. M. Goldmann and M. Karpinski. Simulating threshold circuits by majority circuits. In *STOC*, 1993.
GSW26. Aarushi Goel, Swagata Sasmal, and Mingyuan Wang. Weighted cryptography with weight-independent complexity. Cryptology ePrint Archive, Paper 2026/273, 2026.
Hås94. Johan Håstad. On the size of weights for threshold gates. *SIAM J. on Discrete Mathematics*, 1994.
Hof92. T. Hofmeister. The power of negative thinking in constructing threshold circuits for addition. In *SCTC*, 1992.
Kra94. H. Krawczyk. Secret sharing made short. In *CRYPTO*, 1994.
KRDO17. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO*, 2017.
Lab17. Protocol Labs. Filecoin: A decentralized storage network. technical report, 2017.
Lab22. Aptos Labs. The aptos blockchain: Safe, scalable, and upgradeable web3 infrastructure, 2022.
Mig83. Maurice Mignotte. How to share a secret. In *Cryptography*, 1983.
Mur71. Saburo Muroga. *Threshold Logic and Its Applications*. Wiley-Interscience, 1971.
Pat90. M. S. Paterson. Improved sorting networks with o(logn) depth. *Algorithmica*, 1990.
PS00. Carles Padró and Germán Sáez. Secret sharing schemes with bipartite access structure. *IEEE Transactions on Information Theory*, 2000.
Raz87. Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 1987.
RYS+20. Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability, 2020.
Sha79. Adi Shamir. How to share a secret. *Communications of the ACM*, 1979.
Sol. Solana stake distribution. `https://solscan.io/analytics`. Accessed: 2025-09-15.
TF24. Andrei Tonkikh and Luciano Freitas. Swiper: a new paradigm for efficient weighted distributed protocols. In *PODC*, 2024.
Val84. L. G. Valiant. Short monotone formulae for the majority function. *J. of Algorithms*, 1984.
Yak17. Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain, 2017.
Yao89. A. C. Yao. Unpublished manuscript, 1989. Presented at Oberwolfach and DIMACS workshops.

# A General Version of Yao's Compiler Supporting Threshold Gates

In this section, we briefly describe the idea behind Theorem 2.16, which is a general version of Yao's compiler [Yao89] that supports threshold gates. This is a well-known folklore result in the area of computational secret sharing, whose formal statement has only recently been presented by Beimel in [Bei25]. First, we outline how Yao's compiler [Yao89] is derived from the Benaloh and Leichter [BL88] compiler for monotone Boolean functions with AND and OR gates. Then, we explain how this approach can be extended to handle threshold gates.

Yao's compiler extends the information-theoretic compiler of Benaloh and Leichter from monotone Boolean formulas to monotone circuits by introducing computational assumptions. While it treats AND and OR gates in the same way as Benaloh and Leichter's compiler, it introduces an additional mechanism to handle the multiple fan-out present in monotone circuits. This adaptation relies on the existence of one-way functions. Specifically, for any gate $G$, Yao's compiler uses a semantically-secure symmetric encryption scheme: all outgoing wires of $G$ are encrypted using the same encryption key, and the resulting ciphertexts (one for each fan-out) are made public. Then, the only value that is secret-shared through $G$ is the encryption key itself. Importantly, this additional step does not alter the way AND and OR gates are realized, it only affects the values being propagated. For a more detailed explanation and a formal proof of Yao's compiler, we refer to [Bei25].

Benaloh and Leichter observed that their compiler for constructing secret sharing schemes from monotone Boolean functions can be naturally extended to handle threshold gates by using Shamir's secret sharing scheme. This is possible because Shamir's scheme realizes threshold gates while ensuring that the share size matches the secret size, as long as the secret is defined over a field larger than the number of parties. To extend Yao's compiler to threshold gates, we adopt the same strategy: we apply Shamir's scheme to each threshold gate. By combining this with the original mechanisms for handling AND and OR gates, as well as the previously discussed technique for managing multiple fan-outs in monotone circuits, we obtain a generalized version of Yao's compiler that supports monotone majority Boolean circuits. In fact, as explained in [Bei25], Yao's compiler can be further generalized to work for any monotone Boolean circuit whose gates admit secret sharing schemes with polynomial share size.

# B Illustrative Examples of the Circuits of Section 3

In this section, we provide concrete instantiations of the circuits of Section 3 for $f(\boldsymbol{x}) = \text{sign}(4x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} - 8)$.

## B.1 Monotone Majority Boolean Circuit of Section 3.5

Fig. 6 illustrates the instantiation of the circuit presented in Fig. 3.5 for $f(\boldsymbol{x}) = \text{sign}(4x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} - 8)$.

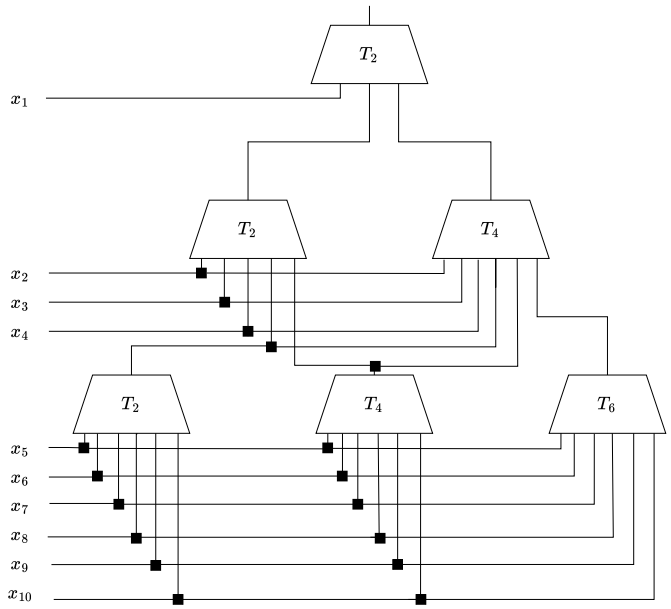


Fig. 6: Monotone majority Boolean circuit realizing the WTAS given by $f(\boldsymbol{x}) = \text{sign}(4x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} - 8)$.

### B.2 Monotone Majority Boolean Circuit of Section 3.7

Fig. 7 illustrates the optimization presented in Section 3.3 for the circuit of Fig. 6.

## C Information-Theoretic Weighted Threshold Schemes from Monotone Formulas

In this section, we further transform our monotone Boolean circuit into a monotone Boolean formula in order to construct an information-theoretic secret sharing scheme with share size $n^{O(\log \log \sigma)}$, matching the best-known result [BW06]. The transformation proceeds in three steps: first, we reduce the depth of our circuit to be logarithmic in the number of parties; then, we convert the resulting circuit into a monotone formula of quasipolynomial size; finally, we apply the compiler by Benaloh and Leichter [BL88] to derive the desired information-theoretic scheme.

The depth of our polynomial-size monotone majority Boolean circuit is $O(\log \sigma)$. Hence, in the worst case, the depth is superlinear in the number of parties. However, since it is an improvement and simplification of the proposal of Beimel and Weinreb [BW06], we can shorten its depth by performing the same balancing and dynamic techniques as in their work.

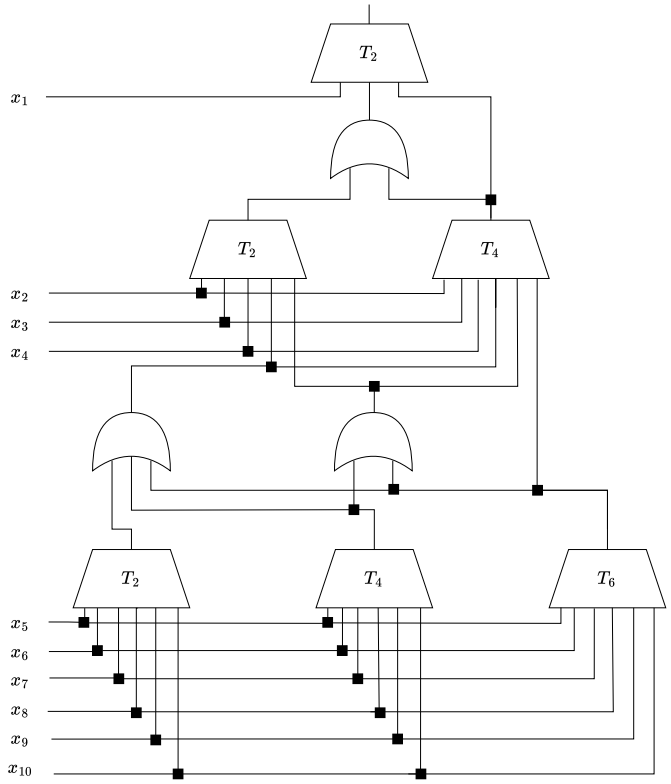


Fig. 7: Optimized version of the circuit realizing the WTAS given by $f(\boldsymbol{x}) = \text{sign}(4x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} - 8)$.

In more detail, the circuit depth can be reduced using a standard balancing technique. This involves computing additions across multiple levels in parallel, using several candidate values for the carry, and determining the correct output with a small selector circuit. Although this process increases the circuit's size, it is subsequently reduced to a polynomial size by properly wiring it using dynamic programming techniques. As a consequence, any monotone WTF over $n$ variables can be computed by a monotone majority Boolean circuit with $\text{poly}(n)$ size and $O(\log \log \sigma)$ depth. For complete details, we refer to the work of Beimel and Weinreb [BW06].

Next, we apply a black-box transformation that converts circuits into formulas to this circuit. This yields a monotone Boolean formula of quasipolynomial size. We then employ the compiler by Benaloh and Leichter [BL88], which constructs information-theoretic secret sharing schemes from monotone formulas. These steps collectively lead to the fact that any WTAS over $n$ parties and

threshold $\sigma$ admits an information-theoretic secret sharing scheme with share size $n^{O(\log\log\sigma)}$. This bound coincides with the one in [BW06]. The statement follows.

**Theorem C.1.** *Any weighted threshold access structure over $n$ parties with threshold $\sigma$ admits an information-theoretic secret sharing scheme with share size $n^{O(\log\log\sigma)}$.*

*Remark C.2.* In the worst case, i.e. when $\log\sigma = O(n\log n)$, the polynomial-size logarithmic-depth monotone majority Boolean circuit we obtain via these balancing and dynamic programming techniques is equivalent to the one presented in the work of Chen, Oliveira and Servedio [COS17]. In this sense, although the depth of the resulting circuit is smaller than that of Section 3.2, they show that its size is larger, that is, of the order of $O(n^{18}\log^6(n))$. Therefore, applying Yao's compiler to it would lead to a computational secret sharing scheme with larger public information than that of Section 3.2.

# D Proofs of Results in Section 3

## D.1 Proof of Theorem 3.2

*Proof.* Without loss of generality, we can assume that the threshold $\sigma$ is greater than any of the weights. Therefore, each weight $w_i$ can be expressed as a sum of at most $\tau = \lceil\log\sigma\rceil$ distinct powers of two. From there, for any $i \in [n]$ let $S_i \subseteq [\tau]_0$ be the set such that $w_i = \sum_{j\in S_i} 2^j$, and let $S_\sigma \subseteq [\tau]_0$ be the set such that $\sigma = \sum_{j\in S_\sigma} 2^j$. We construct $g$ and $\rho$ as follows:

1. Let $k$ be the smallest integer such that $\left\lfloor \frac{\sigma-1}{2^k} \right\rfloor = 0$. Set the threshold $\sigma'$ of $g$ to $2^k$.[4]
2. Define $\delta = \sigma' - \sigma$. If $\delta \neq 0$, let $S_\delta \subseteq [\tau]_0$ be the set such that $\delta = \sum_{j\in S_\delta} 2^j$. Otherwise, let $S_\delta = \emptyset$.
3. Let $m = \sum_{i=1}^n |S_i| + |S_\delta|$. Define a vector of weights $\boldsymbol{v} \in \mathbb{N}^m$ as follows:
   (a) For each variable $x_i$ with weight $w_i$ in $f$, define a set of $|S_i|$ weights $v_{i,j} = 2^j$ for $j \in S_i$.
   (b) Define a set of $|S_\delta|$ weights $v_{\delta,j} = 2^j$ for $j \in S_\delta$. [5]
4. For any $\boldsymbol{y} \in \{0,1\}^m$, define $g'(\boldsymbol{y}) = \mathrm{WTF}(\boldsymbol{v}, \sigma')(\boldsymbol{y})$.
5. Define the mapping $\rho' \colon \{0,1\}^n \to \{0,1\}^m$ such that $y_{i,j} = 1$ if and only if $x_i = 1$ or $i = \delta$.
6. Finally, reorder the weights in $g'$ and the corresponding entries in $\rho'$ to ensure that all weights appear in decreasing order. Denote the resulting function and mapping as $g$ and $\rho$, respectively.

From there, $\rho$ is injective by definition and it is straightforward to check that $f(\boldsymbol{x}) = 1$ if and only if $g(\rho(\boldsymbol{x})) = 1$. In addition, the number of coordinates of $\boldsymbol{v} \in \{0,1\}^m$ is $m = \sum_{i=1}^n |S_i| + |S_\delta| \leq (n+1)\tau = O(n\log\sigma)$.

[4] In this step, $\sigma'$ is obtained by rounding $\sigma$ up to the nearest power of two.

[5] In this step, we add additional weights to account for the increase in the threshold from $\sigma$ to $\sigma'$.

### D.2 Proof of Theorem 3.5

*Proof.* Since $\lambda = \Omega(\log n)$, we apply the general version of Yao's compiler (Theorem 2.16) to the monotone majority Boolean circuit of Theorem 3.3 to obtain the desired computational secret sharing scheme. In this construction, the size of the public information is given by the product of the number of wires and the amount of data transmitted through each wire, that is, $O(\lambda n^2 \log \sigma)$. The share size is equal to the security parameter $\lambda$, which is polynomial or polylogarithmic in the number of parties $n$, depending on the security assumption on the one-way functions.

### D.3 Proof of Theorem 3.7

*Proof.* Let $f$ be a monotone WTF over $n$ variables and threshold $\sigma$, and let $C$ be the monotone majority Boolean circuit of Theorem 3.3 computing it. Due to the general version of Yao's compiler (Theorem 2.16), to prove the result it suffices to build another circuit $C'$ that computes $f$ with the same complexity as $C$ and then argue that the amount of wires needed for any authorized subset to recover the secret during the reconstruction phase is $O(n \log \sigma)$.

To construct the circuit $C'$, we take the circuit $C$ as the starting point. For any level $i \in [O(\log \sigma)]$, we first connect the output wire of each threshold gate $G$ to an OR gate $G'$. Then, each of these OR gates $G'$ also takes as input the output wires of all threshold gates at level $i$ with thresholds higher than that of $G$. Finally, the output wire of $G'$ replaces the original output wire of $G$ as input to all the gates at level $i + 1$.

Once the circuit $C'$ is built, we check its correctness. To do so, it suffices to notice that the output of any of those new OR gates is set to 1 if and only if the output of the smallest threshold gate connected to it is set to 1. In fact, this holds because of the correctness of the circuit $C$.

Now, we move to prove the bound on the number of gates and wires of the circuit $C'$. By construction, it is clear that its depth is twice the depth of $C$, i.e., it is upper bounded by $2 \log \sigma' \leq 2(\log \sigma + 1) = O(\log \sigma)$. Moreover, there are at most $n - 1$ new OR gates and the same amount of threshold gates per level. Finally, each of these new OR gates has at most $n$ input wires, and each threshold gate still has at most $2n$ input wires. This results in upper bounds of $4n \log \sigma' \leq 4n(\log \sigma + 1) = O(n \log \sigma)$ gates and $4n^2 \log \sigma' \leq 4n^2(\log \sigma + 1) = O(n^2 \log \sigma)$ wires for $C'$.

To conclude, it only remains to prove that the amount of wires that each authorized subset needs to use during the reconstruction phase is $O(n \log \sigma)$. By construction, at each level $i \in [2 \log \sigma']$, it suffices to either inspect at most $2n$ input wires of a single threshold gate or a single input wire for at most $n$ OR gates. Therefore, any authorized subset only needs to look at most to $2n$ input and wires per level. This gives a total amount of $4n \log \sigma' \leq 4n(\log \sigma + 1) = O(n \log \sigma)$ wires.

## E Parameters for the Evaluation in Section 4

| **Work** | **Share Size** | **Public Information Size** | **Downloaded Public Information Size** | **Total Share Size** |
|---|---|---|---|---|
| Theorem 1.1 | $\lambda$ | $4\lambda n^2(\log\sigma+1)$ | $4\lambda n(\log\sigma+1)$ | Share Size + Downloaded Public Information Size |
| [ABI+23,FG24] | $\lambda$ | $\lambda(n^2+1830n\log n)\log\sigma$ | | |
| [BW06] | $\lambda$ | $\lambda(n^2+(1830n\log n)^2)\log\sigma$ | | |
| [Sha79] | $w\log W$ | 0 | 0 | |

Fig. 8: Formulas used to compute the metrics in Figure 5.

| **Blockchain Network** | $n$ | $W$ |
|---|---|---|
| Filecoin [Lab17,Fil] | 1343 | $1.28\times10^8$ |
| Avalanche [RYS+20,Ava] | 881 | $1.63\times10^8$ |
| Solana [Yak17,Sol] | 1016 | $5.43\times10^8$ |
| Aptos [Lab22,Apt] | 151 | $8.83\times10^8$ |

Fig. 9: Parameters used to compute the metrics in Figure 5.