From Degree Three to Degree Two: Statistical Randomized Encodings for All Functions

Emmanuela Orsini* Arpita Patra†

Abstract

A randomized encoding represents a function by a simpler randomized map whose output allows recovery of the function value while revealing essentially no further information about the input. Perfectly correct and perfectly private encodings of degree three over $\mathbb{F}_2$ are known for every Boolean function, but degree two cannot provide these guarantees for all functions (Ishai and Kushilevitz, FOCS 2000 and ICALP 2002). Whether degree two suffices when small statistical errors are allowed in both correctness and privacy has remained open.

We construct quadratic randomized encodings of every Boolean function with exponentially small correctness and statistical privacy errors. The degree bound holds jointly in the input bits and independent uniform random bits, and the construction, encoder, decoder and simulator have complexity polynomial in the total size of mod-2 branching programs computing the function's output bits and in the security parameter. Our encodings also achieve decomposability and output locality three, meaning that each encoded bit depends on at most three input and random bits. These results resolve the degree-two and locality-three questions for statistical randomized encodings (Ishai, 2013). As an application, we obtain one-way functions of optimal output locality three from one-way functions in polynomial-time uniform NC^1, improving the previous locality-four result under the same assumption (Applebaum, Ishai and Kushilevitz, FOCS 2004).

We also obtain quadratic multiparty randomized encodings with negligible correctness and statistical privacy errors against semi-honest corruption of any coalition of parties. Previous general constructions for $k \geq 4$ parties provided perfect correctness and privacy against at most $\lfloor 2k/3 \rfloor$ corruptions (Applebaum, Ishai, Karni and Patra, CRYPTO 2022). Through known reductions, our encodings yield non-interactive reductions of every functionality to a fixed three-party functionality and three-round protocols in the two-round OT-hybrid model, with negligible correctness and statistical privacy errors against every coalition. They also yield non-interactive client–server protocols for every function whenever a majority of the servers is honest, even when corrupted servers collude with any set of clients. This server threshold is optimal in this model when privacy is required against arbitrary client coalitions.

*Bocconi University, Milan, Italy. emmanuela.orsini@unibocconi.it

†Indian Institute of Science, Bangalore, India. arpita@iisc.ac.in

1 Introduction

A randomized encoding represents a function f by a randomized map $E(x;r)$ whose output allows a decoder to recover $f(x)$ while revealing essentially no further information about x. Privacy requires the existence of a randomized simulator that, given only $f(x)$, reproduces the encoding's output distribution exactly under perfect privacy, or within a small statistical distance under statistical privacy. We also allow a small probability of decoding error, with both correctness and privacy guarantees required for every fixed input. The aim is to make the encoder simpler than direct evaluation of f, while keeping decoding efficient.

Introduced for round-efficient secure computation [IK00, IK02], randomized encodings have since found applications to locally computable cryptographic primitives [AIK06b, AIK06a], verifiable delegation [AIK10] and non-interactive secure computation [IKO+11]. We refer to [Ish13, App17] for surveys of their constructions and applications.

A fundamental question in the study of randomized encodings is how low their algebraic degree can be for arbitrary Boolean functions, with degree measured jointly in the input bits and the independent uniform bits of the random tape. Over $\mathbb{F}_2$, degree three suffices for perfectly correct and perfectly private encodings of every Boolean function, with complexity polynomial in the size of a mod-2 branching-program representation of f [IK00, IK02, AIK06b]. When the degree is reduced to two, however, some functions no longer admit perfectly private encodings, even if we allow a correctness error $\delta < 1/2$ [IK00]. A related measure is output locality, the maximum number of input and random bits on which a single encoded bit depends. Low degree alone does not bound locality: for example, $\sum_{i=1}^{m} x_i r_i$ is quadratic but depends on all $2m$ bits. Nevertheless, the transformation of Applebaum, Ishai and Kushilevitz converts the degree-three encodings into encodings of output locality four [AIK06b].

Whether degree two suffices for every Boolean function under statistical privacy has been a longstanding open question [IK00, Ish13, AHI+17, AIK18, ABG+20, HIKR23]. The problem remained open even when arbitrarily small correctness and privacy errors were allowed, as did the analogous question for output locality three [Ish13].

For applications to secure computation, we also consider multiparty randomized encodings, where inputs are distributed among parties and privacy must hold against static, semi-honest corruptions [ABT18, AIKP22]. In this setting, the parties may perform arbitrary local preprocessing, and the degree of the public polynomial map is measured jointly in the resulting local values and any internal randomness. Every functionality with at most three parties admits a quadratic encoding that is perfectly private against every coalition [AIKP22], but for $k \geq 4$ the same general construction provides perfect correctness and perfect privacy only against coalitions of at most $\lfloor 2k/3 \rfloor$ parties. Whether every functionality admits a quadratic multiparty encoding that is private against every coalition remained open, even with arbitrarily small correctness and statistical privacy errors [ABT18, AIKP22].

1.1 Our results

Our main result shows that every Boolean function admits a randomized encoding of degree at most two (2RE) with arbitrarily small correctness and statistical privacy errors.

Main Theorem (informal) (Quadratic statistical RE; Theorem 4.12)**.** *Every function with a mod-2 branching-program representation of total size σ has, for every integer $\lambda \geq 1$, a randomized encoding of degree at most two over $\mathbb{F}_2$, jointly in its input bits and its independent uniform random bits, with correctness and statistical privacy errors at most $2^{-\lambda}$ on every input. The construction, encoder, decoder and simulator run in time polynomial in $\sigma + \lambda$.*

The construction requires no computational assumptions, and its randomness length, output length and expanded polynomial description size are polynomial in $\sigma + \lambda$. As observed in [Ish13], a statistical 2RE of $g^\star(a, b, c, d) = abc + d$ suffices to obtain such encodings for every Boolean function from the known perfect degree-three encodings [IK00, AIK06b]. Our encoding of $g^\star$ uses $O(\lambda^3)$ random bits and $O(\lambda^3)$ output bits, and its encoder, decoder and simulator run in time $O(\lambda^3)$.

Locality and one-way functions. Applying the locality construction of Applebaum, Ishai and Kushilevitz [AIK06b] to our quadratic encodings gives encodings of degree at most two and output locality three, without increasing either error bound. Each encoded bit therefore depends on at most three input and random bits in total and can be computed by a bounded-fan-in circuit of constant depth.

The resulting encodings are also decomposable, that is, for input bits $x_1, \ldots, x_N$ and random tape R, the encoding can be written as $E(x; R) = \big(E_0(R), E_1(x_1; R), \ldots, E_N(x_N; R)\big)$, where E_0 depends on the randomness alone and E_i depends only on x_i and R. Thus, if different parties hold different input bits and share the random tape, each can compute the blocks corresponding to its own inputs. As an application, we obtain one-way functions of output locality three from one-way functions in polynomial-time uniform NC^1, improving the previous locality-four result of Applebaum, Ishai and Kushilevitz [AIK06b] under the same assumption. Locality three is optimal, and one-way functions with this locality were previously known only under specific assumptions, such as the hardness of decoding random linear codes [AIK06b] or the one-wayness of random local functions [AIK18].

Multiparty randomized encodings. We next use our quadratic randomized encodings to construct quadratic multiparty randomized encodings (2MPREs). To encode a party's output, every party independently samples a random string, and we use the sum of these strings as the encoder's random tape. Each party computes local values from its own input and randomness, and a public map of degree at most two combines these values to produce the encoded outputs.

Main Theorem (informal) (Quadratic statistical MPREs; Corollary 6.2)**.** *Every deterministic k-party functionality with bit-string inputs and outputs and a mod-2 branching-program representation of total size σ has, for every integer $\lambda \geq 1$, a multiparty randomized encoding over $\mathbb{F}_2$ of effective degree at most two, without internal randomness, with correctness error at most $2^{-\lambda}$ for each party and statistical privacy error at most $2^{-\lambda}$ against every coalition. Its construction, local maps, public map, decoders and simulator run in time polynomial in $k + \sigma + \lambda$.*

Our result allows coalitions of any size by allowing negligible correctness and privacy errors, and thus answers the question of [ABT18] in the statistical setting.

Three-party completeness and three-round protocols. Our fully private statistical multiparty encodings give non-interactive reductions of every deterministic functionality to parallel calls to the fixed three-party functionality TOT, using the reductions of [ABG+20, AIKP22]. Here TOT takes $(a, u) \in \mathbb{F}_2^2$ from one party and $(b, v) \in \mathbb{F}_2^2$ from another, and delivers $ab + u + v$ only to a third party, which has no input. Implementing these calls in the two-round OT-hybrid model gives three-round protocols for the original functionality. Both transformations preserve the correctness and statistical privacy error bounds, with privacy against every coalition. For public-output functionalities, the same protocols also provide statistical privacy against the weakly-active adversaries of [AIKP22], with correctness required only in passive executions. We state the precise guarantees in Sections 6.3.1 and 6.3.2.

Client–server computation. Our main application uses these multiparty encodings to obtain non-interactive client–server protocols [BIW10, ABT18, AIKP22]. The input clients hold the inputs to the functionality, while an additional client, the designated receiver, is the only party prescribed an output. The receiver and the n servers have no inputs to the functionality. Each client, including the receiver, sends one private message to each server, and each server then sends one private reply to the receiver, who computes its output from these replies and its local randomness.

For every deterministic functionality with this output assignment, our construction gives a non-interactive client–server protocol whenever $n > 2t$, where at most t servers may be corrupted. For every integer $\lambda \geq 1$, the protocol has correctness and statistical privacy errors at most $2^{-\lambda}$ against static, semi-honest corruptions, even when the corrupted servers collude with any set of clients, including the receiver. For $k \geq 4$ clients, including the receiver, previous work provided privacy against coalitions containing at most $\lfloor 2k/3 \rfloor$ clients under the same condition on the servers [AIKP22]. The question of tolerating every coalition of clients was raised in [AIKP22], following [BIW10]. For general statistical computation with privacy against arbitrary client coalitions, the condition $n > 2t$ is optimal in this model: when $n \leq 2t$, the classical partition argument shows that some functionalities cannot be computed with both errors negligible [BGW88, CK91].

Computational variants. We also obtain computationally private encodings whose complexity is polynomial in the circuit size and security parameter, and in the number of parties in the multiparty setting. For ordinary randomized encodings, we use the *easy PRG* assumption of Applebaum, Ishai and Kushilevitz [AIK06a]. This requires a secure pseudorandom generator whose output bits can each be computed from the seed by a mod-2 branching program, with both the program size and its construction time polynomial in the seed length. Using our quadratic encoding of $g^\star = abc + d$, we reduce the degree of their computationally private encodings from three to two, preserving computational privacy and introducing negligible correctness error.

For multiparty randomized encodings, we combine our fully private quadratic encodings with the known circuit reductions of [ABG+20, AIKP22]. These reductions evaluate the generator during the parties' local preprocessing, which is not subject to the degree bound, so any secure PRG suffices. The resulting multiparty encodings have effective degree at most two, negligible correctness error and computational privacy against every coalition.

Additional applications. Our decomposable statistical encodings of output locality three have further applications such as single-server instance hiding with quadratic query maps, private simultaneous-message (PSM) and conditional-disclosure of secrets (CDS) protocols whose message bits have locality at most three, and single-input private garbling, all with negligible correctness and statistical privacy errors. They also allow existing cryptographic message-generation algorithms to be replaced by maps of output locality at most three. Under the easy PRG assumption, the same transformation has polynomial overhead in circuit size and the security parameter, with the security properties preserved in each case specified in Section 5.4.

Table 1 compares our main results with the previous constructions.

1.2 Related work

A related line of work studies additive randomized encodings [HIKR23]. To encode a function $f(x_1, \ldots, x_k)$, each party P_i uses independent randomness r_i to compute a group element $\mathrm{Enc}_i(x_i; r_i) \in G$, where G is an Abelian group. The decoder receives only the sum $\sum_{i=1}^{k} \mathrm{Enc}_i(x_i; r_i)$, from which it recovers the function value, while privacy requires that this sum reveal essentially no further information about the inputs. Statistical constructions for all functions were obtained

Result and assumptions	Previous constructions	This paper
REs, mod-2 branching programs	Degree three; perfect [IK00, AIK06b]	Degree two; statistical
Decomposable REs, mod-2 branching programs	Locality four; perfect [AIK06b]	Degree two; locality three; statistical
REs, circuits; easy PRG	Degree three; computational privacy [AIK06a]	Degree two; computational privacy
k-party 2MPREs, mod-2 branching programs	For $k \geq 4$: perfect; at most $\lfloor 2k/3 \rfloor$ corruptions [AIKP22]	Statistical; every coalition
k-party 2MPREs, circuits; any secure PRG	For $k \geq 4$: computational privacy; at most $\lfloor 2k/3 \rfloor$ corruptions [AIKP22]	Computational privacy; every coalition
OWFs from OWFs in polynomial-time uniform NC^1	Locality four [AIK06b]	Degree two; locality three (optimal)
Non-interactive client–server protocols; k clients including the receiver; n servers, at most t corrupted	For $n > 2t$ and $k \geq 4$: perfect; at most $\lfloor 2k/3 \rfloor$ corrupted clients [AIKP22]. For $n > t$: statistical; honest input clients, possibly corrupted receiver [BEGI26]	For $n > 2t$: statistical; every client coalition

Table 1: Comparison with previous constructions. "Perfect" refers to both correctness and privacy; "statistical" allows errors at most $2^{-\lambda}$ in both. In the computational rows, previous constructions have perfect correctness, while ours have negligible correctness error.

in [BEGI26]. The sum itself defines an ordinary randomized encoding of the full input $(x_1, \ldots, x_k)$, but the additive structure does not bound the degree of the maps Enc_i in their input and random bits, and addition in G need not be addition over $\mathbb{F}_2$. These results therefore do not by themselves yield quadratic randomized encodings over $\mathbb{F}_2$, where degree is measured jointly in the original input bits and the independent uniform random bits. The construction of [BEGI26] also gives non-interactive client–server protocols with statistical security against any strict subset of the servers, even when the receiver is corrupted, provided that all input clients are honest. Our protocols instead allow arbitrary coalitions of input clients to collude with the receiver and corrupted servers, under the condition $n > 2t$. The two results therefore cover different corruption guarantees, and our optimality claim for the server threshold is made under the latter requirement.

Our constructions allow a small correctness error and therefore leave open whether every Boolean function admits a degree-two randomized encoding with perfect correctness and statistical privacy. This question was raised in [AHI+17] in connection with low-degree collision-resistant hashing and statistically hiding commitments. Our results also do not yield pseudorandom generators or collision-resistant hash functions of output locality three.

Organization. Section 2 presents a detailed overview of our main constructions and proofs; Section 3 gives the basic notation and definitions; Section 4 develops the quadratic encodings and their computational variant for circuits; Section 5 derives locality three and the applications to one-way functions, single-server instance hiding, shared-randomness protocols and cryptographic message generation; Section 6 develops the multiparty encodings and derives reductions to three-party functionalities, three-round protocols in the two-round OT-hybrid model, weakly-active privacy and client–server protocols.

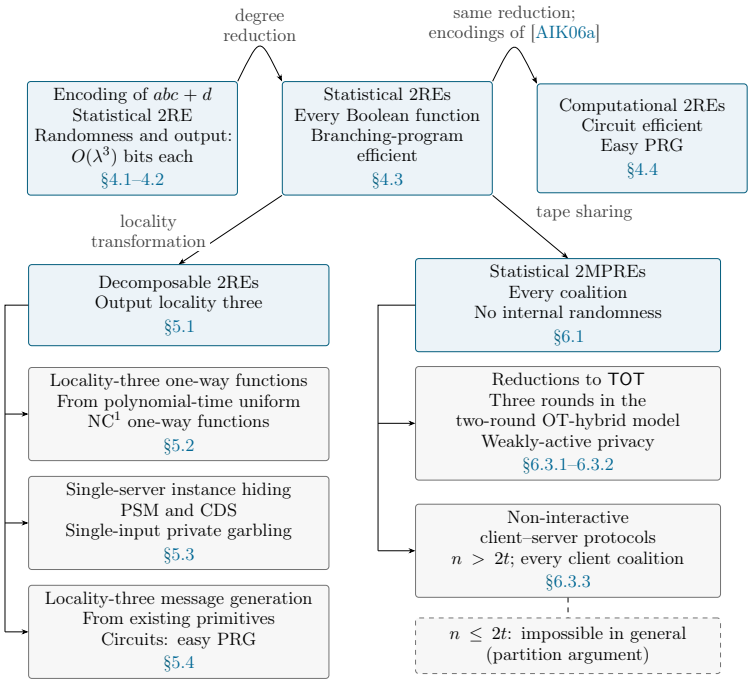


Figure 1: The quadratic encodings and their applications.

2 Technical Overview

Figure 1 summarizes the connections between our constructions and their applications. We start with our quadratic encoding of $g^\star(a, b, c, d) = abc + d$ and its analysis, then explain how it yields encodings of arbitrary functions, output locality three and fully private multiparty encodings.

2.1 Our degree-two randomized encoding

We work over $\mathbb{F}_2$ and start from the known perfect encodings of degree three, in which every output coordinate is a sum of products of at most three input or random bits [IK00, IK02, AIK06b]. To reduce the degree to two, we would like to encode each of these products separately and let the decoder recover every coordinate by adding the decoded values. Since the individual products may reveal more than their sum, we first mask them choosing the masks uniformly subject to summing to zero within each coordinate. This leaves the sum unchanged and ensures that the masked products reveal only the original coordinate. Each masked product is a value of the function $g^\star(a, b, c, d) = abc + d$, where a, b, c are its factors with missing factors set to 1, and d is its mask. Therefore, as observed in [Ish13], a quadratic encoding of $g^\star$ suffices to obtain quadratic encodings of arbitrary Boolean functions: we recover each original encoded coordinate by adding the decoded masked products, then apply the original decoder to obtain the function value.

2.1.1 Encoding of $abc + d$

Rank testing. Following the matrix-rank approach of Ishai and Kushilevitz [IK00, IK02], we represent the masked product $abc + d$ as a determinant:

$$M(a, b, c, d) = \begin{pmatrix} 1 & 0 & c \\ a & 1 & 0 \\ 0 & b & d \end{pmatrix}, \qquad \det M(a, b, c, d) = abc + d.$$

Since the first two columns are always independent, M has rank two when the answer is zero and rank three when it is one. Its rank therefore determines the desired output, but releasing the matrix would also reveal all four input bits. For this reason, we sample a uniform matrix $N \in \mathbb{F}_2^{h\times 3}$, for a public parameter $h \geq 3$ that we will specify later, and output $Q = NM$. The rows of Q are independent uniform samples from $\mathrm{Row}(M)$, and every entry has degree at most two jointly in the input bits and the independent random bits of N.

Given Q, the decoder computes its rank and returns one if this rank is three, and zero otherwise. To see why this works, we consider the two possible output values separately. When $abc + d = 0$, multiplication by N cannot increase the rank of M, so Q has rank at most two and the decoder is always correct. When $abc + d = 1$, the matrix M is invertible, so $Q = NM$ has the same rank as N and the decoder fails exactly when this rank is less than three. In that case, all h rows of N lie in one of the seven two-dimensional subspaces of $\mathbb{F}_2^3$. Each such subspace contains four of the eight possible row vectors, so a uniform row lies in it with probability $1/2$, and all h independently sampled rows lie in it with probability 2^{-h}. Summing over the seven subspaces therefore bounds the decoding error by $7 \cdot 2^{-h}$, which we can make arbitrarily small by increasing h. Since the decoder is always correct when $abc + d = 0$, this bound holds for every input. Privacy requires that inputs with the same value of $abc + d$ give essentially the same distribution of Q. This holds when $abc + d = 1$, since Q is uniform over $\mathbb{F}_2^{h\times 3}$, but when $abc + d = 0$ we have $d = abc$, so the third row of M is a linear combination of the first two, which depend only on a and c. For our choice of M, its row space is therefore $\mathrm{Row}(M) = \{(z_1, z_2, z_3) \in \mathbb{F}_2^3 : z_3 = cz_1 + acz_2\}$, and the rows of Q are independent uniform samples from this plane. The three possible pairs (c, ac), i.e., $(0, 0), (1, 0), (1, 1)$, give distinct planes. An observer can identify the plane whenever the sampled rows span it. Indeed, a two-dimensional space over $\mathbb{F}_2$ has three one-dimensional subspaces, and the probability that all h rows lie in any fixed one is 2^{-h}. By a union bound, the sampled rows span the plane with probability at least $1 - 3 \cdot 2^{-h}$. Consequently, Q reveals c, and also a when $c = 1$, with probability at least $1 - 3 \cdot 2^{-h}$, even though the prescribed output is zero in every case. So increasing h improves correctness but does not resolve the privacy problem.

To reduce this leakage, we use two recursive linear secret-sharing schemes for a and c, together with an additive sharing of d, and apply the matrix construction above to corresponding shares. For either recursive sharing, every fixed set of shares either determines the shared bit or reveals nothing about it. We will use this property to bound the privacy error in terms of the probabilities that selected shares determine a or c.

The two recursive sharings. We use different sharing rules for a and c because the matrix distribution depends on them differently: when the encoded value is zero, the plane always determines c, but determines a only when $c = 1$. The two sharing rules must ensure that the shares exposed by the rank encodings rarely suffice to reconstruct either secret, while allowing the product ac to be recovered by multiplying corresponding shares of a and c and adding the results. We first describe the two linear secret-sharing schemes that we use to replace each input bit by four shares, as shown in Figure 2, and then apply these schemes recursively. Concretely, their one-level sharing rules are

$$\mathsf{Sh}_A(a; \theta) = (a + \theta, a + \theta, \theta, \theta), \qquad \mathsf{Sh}_C(c; \theta_1, \theta_2) = (c + \theta_1, \theta_1, c + \theta_2, \theta_2),$$

where $\theta, \theta_1, \theta_2$ are independent uniform bits. The two schemes differ in which sets of shares allow reconstruction. For Sh_A, an authorized set must contain a share from each of the two groups $\{1, 2\}$ and $\{3, 4\}$, because adding these shares gives $(a + \theta) + \theta = a$. If the available shares are restricted to either group, they are only repeated copies of a uniform bit, whose distribution is independent of a. For Sh_C, either complete pair $\{1, 2\}$ or $\{3, 4\}$ is enough to reconstruct c, since $(c + \theta_r) + \theta_r = c$, with $r \in \{1, 2\}$. If neither pair is complete, the available shares contain at most one bit from each

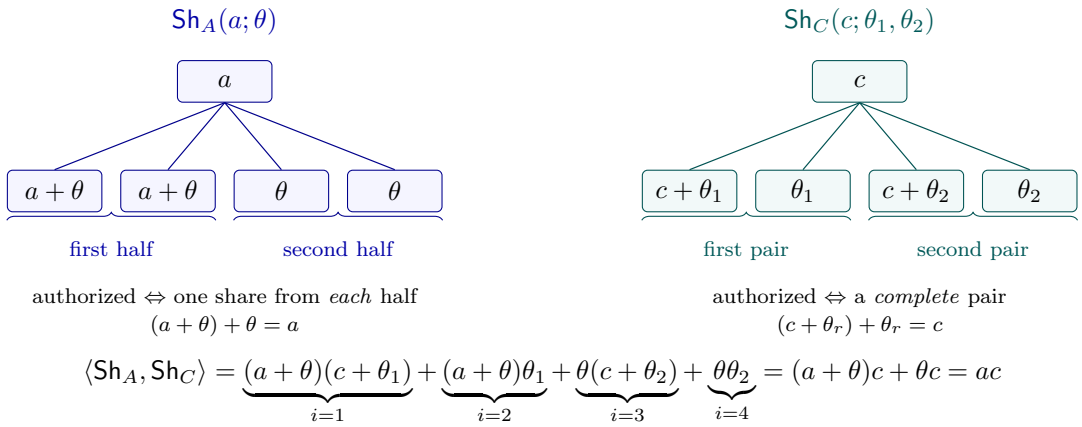


Figure 2: The one-level sharing rules. Reconstruction in Sh_A requires a share from each half, while reconstruction in Sh_C requires a complete pair. Adding the products of corresponding shares gives ac.

independently masked pair and, since these bits are independent and uniform, their joint distribution is independent of c. Thus, in both schemes, an authorized set reconstructs the secret exactly, whereas an unauthorized set reveals nothing about it. Notice that the schemes we just described are instances of the linear secret-sharing construction of Benaloh and Leichter [BL90]. The reason for using these two schemes together is that multiplying all the corresponding shares and adding the four products recovers ac for every choice of masks:

$$\big\langle \mathsf{Sh}_A(a;\theta), \mathsf{Sh}_C(c;\theta_1,\theta_2)\big\rangle = (a+\theta)(c+\theta_1) + (a+\theta)\theta_1 + \theta(c+\theta_2) + \theta\theta_2 = (a+\theta)c+\theta c = ac.$$

Thus, each scheme hides its secret from unauthorized sets of shares, while the products of corresponding shares still sum to ac.

We now repeat each sharing rule for j levels. Starting with a, we apply Sh_A to obtain four shares, then apply Sh_A separately to each of these shares, and continue in the same way; we do the same for c using Sh_C. Every instantiation uses fresh independent masks, including when two shares have the same value. Each procedure forms a tree with four children per node, whose $L = 4^j$ leaves give the final share vectors $\mathbf{a} = (a_1, \ldots, a_L) = \mathsf{Sh}_A^j(a; \boldsymbol{\theta}_A)$ and $\mathbf{c} = (c_1, \ldots, c_L) = \mathsf{Sh}_C^j(c; \boldsymbol{\theta}_C)$, where $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_C$ collect all the masks used in the two trees. We list the leaves in the same order, so a_i and c_i correspond to the same sequence of child indices.

At each pair of corresponding nodes, the one-level identity says that the product of their values equals the sum of the products of their four corresponding child shares. Summing these identities over all pairs at a given level shows that the sum of corresponding products is unchanged at the next level. Since this sum is ac at the roots, after j levels we obtain $\sum_{i=1}^{L} a_i c_i = ac$.

In this way, using the same bit b at every leaf gives $\sum_i a_i b c_i = abc$, which will allow the decoder to recover the original cubic term from the leaf values. Moreover, every leaf share is a linear combination of its original input bit and the sharing masks, so substituting these shares into the matrix construction preserves the quadratic degree bound.

The additive sharing and the random matrices. We share d additively across the same leaves by choosing $\mathbf{d} = (d_1, \ldots, d_L)$ uniformly subject to $\sum_i d_i = d$ and independently of both recursive sharings. At leaf i, the shares a_i, c_i, d_i, together with the original bit b, define the value $y_i = a_i b c_i + d_i$.

The same b is used at every leaf, so the product and additive-sharing identities give

$$\sum_{i=1}^{L} y_i = b \sum_{i=1}^{L} a_i c_i + \sum_{i=1}^{L} d_i = abc + d.$$

Once the leaf values are recovered, adding them modulo two, or equivalently taking their parity, therefore gives the desired output.

For the privacy analysis, we will use the fact that, for every fixed input, $\mathbf{y}$ is uniform among all vectors with parity $abc + d$ and is independent of the masks used in the two recursive sharings. We can therefore generate the same joint distribution by sampling the two recursive sharings and independently choosing $\mathbf{y}$ subject only to this parity constraint.

At this point, to conclude the encding, we sample independent uniform matrices $N_1, \ldots, N_L \in \mathbb{F}_2^{h \times 3}$ and use N_i for the rank encoding at leaf i.

Encoding and decoding. Using these shares and matrices, the encoder outputs

$$E_{j,h}(a, b, c, d; R) = (Q_1, \ldots, Q_L), \qquad Q_i = N_i \cdot \begin{pmatrix} 1 & 0 & c_i \\ a_i & 1 & 0 \\ 0 & b & d_i \end{pmatrix},$$

where R collects all the independent random bits used in the sharings and coefficient matrices N_i. To see directly why the complete encoder is quadratic, consider a row (n_1, n_2, n_3) of N_i. Its corresponding output row is $(n_1 + n_2 a_i,\ n_2 + n_3 b,\ n_1 c_i + n_3 d_i)$. Each share is affine in the original inputs and sharing masks, so multiplying it by a fresh coefficient bit gives degree at most two.

The decoder tests the rank of each matrix and returns the parity of the resulting bits:

$$\mathsf{Dec}_{j,h}(Q_1, \ldots, Q_L) = \sum_{i=1}^{L} \mathbf{1}\{\mathrm{rank}(Q_i) = 3\}.$$

Whenever all rank tests recover their leaf values correctly, the decoder returns $\sum_i y_i = abc + d$. As mentioned before, a test is always correct when $y_i = 0$. When $y_i = 1$, it returns an incorrect value with probability at most $7 \cdot 2^{-h}$ over the independent choice of N_i. For $j \geq 1$, each leaf value equals one with probability $1/2$, because the leaf vector is uniform subject only to its fixed parity. The error probability at any given leaf is therefore at most $\frac{1}{2} \cdot 7 \cdot 2^{-h} = \frac{7}{2}\, 2^{-h}$, and a union bound over the L leaves gives the following bound.

Proposition (Correctness; Proposition 4.3)**.** *For all integers $j \geq 1$ and $h \geq 3$, with $L = 4^j$, and every input $(a, b, c, d) \in \mathbb{F}_2^4$, $\Pr_R\big[\mathsf{Dec}_{j,h}(E_{j,h}(a, b, c, d; R)) \neq abc + d\big] \leq \frac{7}{2} \cdot L \cdot 2^{-h}$.*

Once the depth j has been chosen to obtain privacy, we can thus increase h to make decoding reliable without changing the degree of the encoding.

Privacy. To prove privacy, we need to define a simulator that, given only the prescribed output $y = abc + d$, reproduces the distribution of the encoded matrices up to a small statistical error. Our simulator runs the encoder on $(0, 0, 0, y)$, which has the required output y, using fresh randomness, i.e., $\mathsf{Sim}_{j,h}(y) = E_{j,h}(0, 0, 0, y; R')$. We prove its privacy guarantee by bounding $\mathrm{SD}\big(E_{j,h}(a, b, c, d; R), \mathsf{Sim}_{j,h}(abc + d)\big)$ uniformly over all inputs $(a, b, c, d) \in \mathbb{F}_2^4$, where R and R' are independent uniform random tapes.

The simulator encodes $(0, 0, 0, y)$, so we must show that its output has almost the same distribution as the encoding of (a, b, c, d). We describe how to sample the rows using only the leaf bits and

selected shares of a and c, so that we can apply the privacy of the recursive sharings. Keeping the leaf bits fixed, we then replace each root by zero: this changes the distribution only when the selected shares can reconstruct that root. Bounding the probability of these events proves privacy, since after both replacements we obtain the simulator's distribution. Concretely, given (y_i, c_i, a_i), the procedure $\mathsf{RowSamp}$ samples fresh independent uniform bits U, V, W on each call and returns

$$\mathsf{RowSamp}(y_i, c_i, a_i) = \begin{cases} (U, V, W), & y_i = 1, \\ (U, V, 0), & y_i = 0,\ c_i = 0, \\ (U, V, U + a_i V), & y_i = 0,\ c_i = 1. \end{cases}$$

By the row-space calculation in the rank-testing analysis, this output has exactly the conditional distribution of a row of Q_i given the shares. Applying $\mathsf{RowSamp}$ with fresh independent randomness for every row therefore reproduces the joint distribution of the encoded matrices. $\mathsf{RowSamp}$ uses c_i only when $y_i = 0$, and a_i only when also $c_i = 1$, so the shares it needs are indexed by $S_C = \{i : y_i = 0\}$ and $S_A = \{i : y_i = 0,\ c_i = 1\}$.

Since the leaf vector is independent of the recursive masks, the matrices admit the following exact sampling description.

Lemma (Sampling from selected shares; Proposition 4.4, Step 1)**.** *Fix (a, b, c, d) and let $y = abc + d$. Sample $\mathbf{a} = \mathsf{Sh}^j_A(a; \boldsymbol{\theta}_A)$ and $\mathbf{c} = \mathsf{Sh}^j_C(c; \boldsymbol{\theta}_C)$ with independent uniform masks and, independently, a leaf vector $\mathbf{y}$ uniform on $\mathcal{P}_y = \{\eta \in \mathbb{F}_2^L : \sum_i \eta_i = y\}$. Generate every row of every Q_i using $\mathsf{RowSamp}(y_i, c_i, a_i)$, with fresh independent randomness for every call. The resulting list of matrices has exactly the distribution of $E_{j,h}(a, b, c, d; R)$. In particular, given $\mathbf{y}$, the matrices can be generated using only the selected shares $\mathbf{c}[S_C]$ and $\mathbf{a}[S_A]$.*

Notice that in the sampling procedure above, b and d are used only to specify the parity $y = abc+d$ of $\mathbf{y}$. We therefore compare the real encoding with the simulator by keeping the distribution of $\mathbf{y}$ fixed and replacing the sharings of a and c by sharings of zero, using the following property of linear secret sharing.

Lemma (Lemma 4.6)**.** *For $X \in \{A, C\}$ and every set $S \subseteq [L]$, either a linear map recovers the root s from $\mathsf{Sh}^j_X(s; \boldsymbol{\theta})[S]$ for every s and every $\boldsymbol{\theta}$, in which case we call S authorized, or the distributions of $\mathsf{Sh}^j_X(0; \boldsymbol{\theta})[S]$ and $\mathsf{Sh}^j_X(1; \boldsymbol{\theta})[S]$ over uniform masks are identical.*

For any fixed unauthorized set, Lemma 4.6 guarantees that the shares have exactly the same distribution for secret values zero and one. Since the simulator runs the encoder on $(0, 0, 0, y)$, we use this equality to replace the sharing of a by a sharing of zero and then do the same for c, keeping $\mathbf{y}$ uniform on $\mathcal{P}_y$. We will show that each replacement changes the encoding's distribution by at most the probability that the corresponding selected set is authorized.

To bound these reconstruction probabilities recursively, we first take independent uniform leaf bits $\tilde{y}_1, \ldots, \tilde{y}_L$, without imposing the parity constraint. With $\mathbf{c} = \mathsf{Sh}^j_C(c; \boldsymbol{\theta}_C)$ sampled independently of these leaf bits, we define

$$p_j = \Pr\big[\{i : \tilde{y}_i = 0\} \text{ is authorized for } \mathsf{Sh}^j_C\big],$$
$$q_j(c) = \Pr\big[\{i : \tilde{y}_i = 0,\ c_i = 1\} \text{ is authorized for } \mathsf{Sh}^j_A\big], \qquad q^*_j = \max\{q_j(0), q_j(1)\}.$$

Whether a set is authorized does not depend on the secret being reconstructed, so p_j has no secret argument and $q_j(c)$ does not depend on a; it depends on c only through the rule that selects the a-shares.

Combining the sampling description above with Lemma 4.6 gives the following privacy bound. The factors of two appear because the independent leaf bits have parity y with probability $1/2$, so conditioning on this event increases each authorization probability by at most a factor of two.

Lemma (Privacy error; Lemma 4.7)**.** *For all integers $j \geq 0$ and $h \geq 3$ and every input (a, b, c, d),*

$$\mathrm{SD}\big(E_{j,h}(a, b, c, d; R),\, \mathsf{Sim}_{j,h}(abc + d)\big) \leq 2q_j^* + 2p_j.$$

To prove the lemma, we first replace the A-sharing of a by a sharing of zero, keeping $\mathbf{y}$ and the C-sharing unchanged. Once $\mathbf{y}$ and the selected c-shares are fixed, S_A is fixed and the A-sharing masks remain independent and uniform. By Lemma 4.6, the selected A-shares have the same distribution whenever they do not allow reconstruction, so the change in the matrix distribution is at most the probability that S_A allows reconstruction.

We then replace the C-sharing by a sharing of zero. Since S_C is fixed for each $\mathbf{y}$, the same argument bounds the change in $(\mathbf{y}, \mathbf{c}[S_C])$ by the probability that S_C allows reconstruction. Although this replacement may change S_A, both matrix distributions are obtained from these data by applying $\mathsf{RowSamp}$ with an independent A-sharing of zero. This common randomized procedure cannot increase statistical distance, so the bound also holds for the matrices.

After both replacements, the matrices have distribution $\mathsf{Sim}_{j,h}(y)$. The actual leaf vector is obtained by conditioning independent uniform leaf bits on parity y, an event of probability $1/2$, so each reconstruction probability is at most twice its value without this conditioning. The first replacement therefore changes the distribution by at most $2q_j(c)$, and the second by at most $2p_j$. By the triangle inequality, the distance between the original encoding and the final simulated distribution is at most the sum of these two changes, giving $\mathrm{SD}\big(E_{j,h}(a, b, c, d; R), \mathsf{Sim}_{j,h}(y)\big) \leq 2q_j(c) + 2p_j \leq 2q_j^* + 2p_j$, as required.

Bounding the reconstruction probabilities. We now explain why increasing the sharing depth makes reconstruction unlikely. To recover the root, the selected leaf shares must first recover a suitable combination of its four child secrets. Each child secret is itself shared by a smaller tree, so we can express the probability of recovering the root in terms of the reconstruction probabilities for these smaller trees.

In the C-sharing, reconstructing the root requires reconstructing both child secrets in at least one of the pairs $\{1, 2\}$ or $\{3, 4\}$. The four child reconstruction events are independent, each with probability p_j, so each pair succeeds with probability p_j^2. A union bound over the two pairs gives the first inequality below. For the A-sharing, we need to reconstruct at least one child secret from each pair. Conditional on the two masks at the root of the C-sharing, the child reconstruction events are independent, each with probability at most q_j^*. Each pair therefore succeeds with probability at most $2q_j^*$, and requiring success in both pairs gives the second inequality.

Lemma (Reconstruction probabilities; Lemmas 4.9 and 4.10)**.** *At depth four, $p_4 < 1/10$ and $q_4^* < 1/80$. For every $j \geq 4$,*

$$p_{j+1} \leq 2p_j^2, \qquad q_{j+1}^* \leq 4(q_j^*)^2.$$

The estimates at depth four follow from the exact recurrences in the full analysis and give $2p_4 < 1/5$ and $4q_4^* < 1/20$. Multiplying the two recurrence inequalities by two and four, respectively, gives $2p_{j+1} \leq (2p_j)^2$ and $4q_{j+1}^* \leq (4q_j^*)^2$. Each additional level therefore squares the bounds on these quantities, so after s further levels we obtain $2p_{4+s} \leq (2p_4)^{2^s} \leq 5^{-2^s}$ and $4q_{4+s}^* \leq (4q_4^*)^{2^s} \leq 20^{-2^s}$. Dividing by two and four, respectively, yields $p_{4+s} \leq \frac{1}{2}\, 5^{-2^s}$ and $q_{4+s}^* \leq \frac{1}{4}\, 20^{-2^s}$ for every integer $s \geq 0$.

Parameters. For every integer $\lambda \geq 1$, we choose $j = 4 + \left\lceil \log_2 \left(\frac{\lambda+2}{\log_2 5} \right) \right\rceil$. In addition, we set $L = 4^j$, and choose $h = \lambda + \lceil \log_2(7L) \rceil + 1$. The correctness, privacy and reconstruction bounds then give correctness and privacy errors at most $2^{-\lambda}$. Since $L = O(\lambda^2)$ and $h = O(\lambda)$, the encoding uses $O(\lambda^3)$ random bits and $O(\lambda^3)$ output bits, and its encoder, decoder and simulator run in time $O(\lambda^3)$.

2.1.2 Encoding arbitrary functions

We now apply our quadratic encoding of $abc + d$ to the masked monomials of a perfect degree-three encoding of f, as described at the start of this section, using fresh independent randomness for each copy. The resulting map remains quadratic because each factor is an individual variable or a constant, and the masks are linear in independent random bits.

For privacy, we first simulate the degree-three encoding from $f(x)$ [IK00, AIK06b]. Independently for each simulated coordinate, we then sample the masked monomial values uniformly subject to their sum being that coordinate. This reproduces their joint distribution in the encoding, and we apply the simulator for $abc + d$ to each sampled value using fresh independent randomness.

Let $M \geq 1$ be the total number of encoded monomials, we run each copy of the encoder for $abc + d$ with correctness and statistical privacy errors at most $2^{-\lambda'}$. Summing the corresponding error bounds over the M copies gives at most $M2^{-\lambda'}$ for each error of the complete encoding. We therefore choose $\lambda' = \lambda + \lceil \log_2 M \rceil$, which ensures $M2^{-\lambda'} \leq 2^{-\lambda}$. Since M is polynomial in the branching-program size, the resulting construction has complexity polynomial in that size and λ (Lemma 4.11 and Theorem 4.12).

2.2 Locality three and decomposability

The locality transformation of [AIK06b] gives our encodings output locality three while preserving degree two and decomposability. Moreover, since the transformation has perfect correctness and privacy, neither error increases.

One-way functions. Starting from a one-way function f in polynomial-time uniform NC^1, we would like to obtain a quadratic one-way function by using the map $G(x, r) = E_f(x; r)$, with both x and r treated as inputs. An inverter for G only needs to find a pair (x', r') producing the given encoded string, whereas to invert f we need its first component to satisfy $f(x') = f(x)$. With perfect correctness, this equality would follow by decoding the common encoded string, but our encoding allows decoding errors, so a preimage returned by the inverter may have a different value of f. Although such pairs are rare under uniform sampling, an inverter might consistently choose them, and we therefore cannot directly conclude that G is one-way.

The result of Applebaum, Ishai and Kushilevitz [AIK06b] shows that G is *distributionally one-way*, meaning that it is hard to sample a preimage with approximately the distribution of a uniform input and random tape conditioned on the encoded value. On honestly generated encodings, such a sampler would return pairs with approximately the same distribution as the original uniform pair, and would therefore inherit its small decoding-error probability.

To prove that G is distributionally one-way, we show how a sufficiently accurate preimage sampler for G would allow us to invert f given only $y = f(X)$. Since we do not know X, we generate the sampler's input by running the encoding's simulator $\mathsf{Sim}(y)$, then give the resulting encoded string to the sampler and return the first component X' of the pair it produces. Statistical privacy ensures that replacing an honestly generated encoding by this simulated one changes the probability that $f(X') = y$ by at most the privacy error. The preceding argument therefore implies that an efficient, sufficiently accurate sampler would yield an efficient inverter for f, contradicting its one-wayness.

We then apply the Impagliazzo–Luby conversion [IL89], in the form used by [AIK06b], to turn G into an ordinary one-way function. For each fixed k, the conversion uses the map $(z, A, v) \mapsto (G(z), A, v, Az + v)$, where $z = (x, r)$, A is a uniform binary matrix with k rows and $|z|$ columns, and v is an independent uniform k-bit vector. Including A and v in the output ensures that any returned preimage uses the same coefficients and satisfies the same k linear equations, in addition to reproducing $G(z)$. The construction combines these maps for $k = 1, \ldots, |z|$, using independent inputs for each k, and amplifies hardness by repeating the resulting function on polynomially many independent inputs. The degree remains at most two throughout: the added terms multiply an input bit by a coefficient bit, while combining evaluations on separate inputs introduces no further products. Finally, we apply the locality transformation of Applebaum, Ishai and Kushilevitz [AIK06b] to obtain output locality three, with its perfect correctness and privacy preserving one-wayness.

2.3 Multiparty encodings and their applications

To achieve privacy in the multiparty setting, we share each encoder's randomness among all the parties, using the standard conversion of [Ish13, ABT18]. For a k-party functionality $f = (f_1, \ldots, f_k)$, let E_j be our quadratic encoder for f_j, and $x = (x_1, \ldots, x_k)$ the parties' inputs. Each party P_i independently samples a uniform string $s_{i,j}$ for every $P_j \in [k]$ with the same length as E_j's random tape. We define the public map of the MPRE to take the parties' inputs and these random strings, compute $r^{(j)} = \sum_{i=1}^{k} s_{i,j}$ for each recipient, and produce $E_j(x; r^{(j)})$ as the encoded output delivered only to P_j.

For any proper coalition, a party outside it contributes an independent uniform string to each tape, so the combined tapes remain independent and uniform even given the coalition's own randomness. The simulator therefore samples the coalition's local strings and independently simulates each received encoding from the corresponding prescribed output. If all parties are corrupted, it instead knows the full input and reproduces the execution directly. Replacing each encoder's random tape by a linear sum of the parties' strings preserves degree two, and the public map uses no internal randomness because the parties generate all the randomness themselves. The coalition's privacy error is at most the sum of the privacy errors of the encodings it receives.

Client–server computation. Following [ABT18, AIKP22], we use these multiparty encodings (MPRE) to obtain non-interactive protocols with m input clients, n servers and a designated receiver, where the servers and receiver have no inputs and only the receiver obtains an output. Each client, including the receiver, computes its local MPRE values and shares them among the servers using Shamir secret sharing over a sufficiently large extension of $\mathbb{F}_2$. To tolerate up to t corrupted servers, each value is shared as the constant term of a random polynomial of degree at most t, with each server privately receiving its evaluation at its assigned point among n fixed, distinct nonzero elements of the field. Each server then evaluates the MPRE's quadratic public map on the shares it receives, so the results for each encoded output bit are evaluations of a polynomial of degree at most $2t$ whose constant term is that bit. To hide the other coefficients, each input client also sends evaluations of independent random polynomials of degree at most $2t$ with constant term zero, one per encoded coordinate. The servers add these masks and reply privately to the receiver, who, when $n > 2t$, interpolates the masked polynomials, recovers the encoded output from their constant terms and decodes it.

The protocol preserves both encoding error bounds against static, semi-honest corrupted coalitions containing any set of clients and at most t servers. Under this corruption guarantee, $n > 2t$ is also necessary for general statistical computation with negligible errors, by the classical partition argument [BGW88, CK91] (Lemma A.1).

3 Preliminaries

We use the standard definitions of randomized encodings and their multiparty extension [Ish13, ABT18, AIKP22]. Throughout, δ denotes a correctness error and ε a privacy error. Multiparty privacy is against static semi-honest adversaries.

3.1 Notation

For a nonnegative integer k we write $[k] = \{1, \ldots, k\}$. For a vector $\mathbf{x} = (x_1, \ldots, x_n)$ and a set $T \subseteq [n]$ we write $\mathbf{x}[T] = (x_i)_{i \in T}$. For a prime power q, $\mathbb{F}_q$ is the field with q elements, and $\mathbb{F}$ denotes a finite field, which is $\mathbb{F}_2$ unless stated otherwise. A map $p : \mathbb{F}^a \to \mathbb{F}^b$ is a *polynomial map of degree at most d* if each of its coordinates can be written as a polynomial over $\mathbb{F}$ of total degree at most d in the a variables. Polynomial maps may be specified by arithmetic circuits or by lists of monomials. Whenever a transformation uses a monomial expansion, its cost includes that expansion. For constant d, the expanded description contains at most $b\binom{a+d}{d}$ monomials and therefore has polynomial size in the input and output dimensions.

We work with finite probability spaces. For a finite nonempty set S, U_S denotes the uniform distribution on S, and $s \leftarrow S$ means that s is sampled from this distribution. Similarly, $s \leftarrow \mathcal{D}$ samples s from a distribution $\mathcal{D}$, for instance the output distribution of a randomized algorithm. The *statistical distance* between distributions $\mathcal{D}, \mathcal{D}'$ on a common finite set $\mathcal{Z}$ is

$$\mathrm{SD}(\mathcal{D}, \mathcal{D}') \;=\; \frac{1}{2} \sum_{z \in \mathcal{Z}} \bigl|\mathcal{D}(z) - \mathcal{D}'(z)\bigr| \;=\; \max_{E \subseteq \mathcal{Z}} \bigl(\mathcal{D}(E) - \mathcal{D}'(E)\bigr),$$

where $\mathcal{D}(E) = \sum_{z \in E} \mathcal{D}(z)$. We write $\mathcal{D} \equiv \mathcal{D}'$ if $\mathrm{SD}(\mathcal{D}, \mathcal{D}') = 0$ and $\mathcal{D} \approx_\varepsilon \mathcal{D}'$ if $\mathrm{SD}(\mathcal{D}, \mathcal{D}') \le \varepsilon$. We use the same notation for random variables by identifying them with their distributions.

We use $\lambda \in \mathbb{N}$ for the security parameter. A function $\nu : \mathbb{N} \to [0, 1]$ is *negligible* if, for every constant $c > 0$, we have $\nu(\lambda) < \lambda^{-c}$ for all sufficiently large λ. We write $\mathcal{D}_\lambda \approx_c \mathcal{D}'_\lambda$ when every nonuniform probabilistic polynomial-time distinguisher has negligible distinguishing advantage between the two distributions.

3.2 Randomized encodings

Definition 3.1 (Randomized encodings [ABT18])**.** Let X, Y, R, Z be finite nonempty sets, let $\delta, \varepsilon \in [0, 1]$, and let $g : X \to Y$. A function $\hat{g} : X \times R \to Z$ is a *δ-correct, ε-private randomized encoding* (RE) of g if there are a deterministic decoder $\mathsf{Dec} : Z \to Y \cup \{\perp\}$, where $\perp \notin Y$ denotes failure, and a randomized simulator Sim that maps inputs in Y to outputs in Z, such that, for every $x \in X$,

$$\Pr_{r \leftarrow R}\bigl[\mathsf{Dec}(\hat{g}(x; r)) \neq g(x)\bigr] \le \delta, \qquad \mathsf{Sim}(g(x)) \approx_\varepsilon \hat{g}(x; U_R).$$

The encoding is *perfect* if $\delta = \varepsilon = 0$.

An encoding $\hat{g}$ is useful if it is simpler in some sense than the original function g. In secure computation, a central measure of simplicity is the algebraic degree of the encoding, obtained by expressing its output coordinates as polynomials jointly in the input and random variables and taking their maximum total degree. Other measures of simplicity are useful in other settings; see [Ish13, App17] for surveys on randomized encodings.

We often use E_f to denote an encoding of f, with decoder Dec_f and simulator Sim_f, and omit the security parameter when it is fixed. We use $E_{f,\lambda}$ when the dependence on λ needs to be explicit, and simply E for an arbitrary encoding.

Definition 3.2 (Degree and locality of an RE [AIK06b])**.** An RE $E : \mathbb{F}_2^N \times \mathbb{F}_2^\rho \to \mathbb{F}_2^\ell$ has *degree at most* d if every coordinate admits a polynomial representation of total degree at most d jointly in the original input variables and the independent uniform random bits. We use *2RE* for degree at most two. The RE has *output locality at most* b if each output bit depends on at most b coordinates of the combined vector (x, r).

Efficiency and asymptotic security for both REs and MPREs are specified in Definition 3.7.

We will use the following standard consequences of privacy for every fixed input; see [AIK06b] and [AIK06a].

Lemma 3.3 (Auxiliary information and fresh randomness)**.** *Let $E(z; R)$ be an ε-private RE of $F(z)$ with simulator S, and let (Z, A) be independent of the uniform tape R. Then*

$$\mathrm{SD}\big((Z, A, E(Z; R)), (Z, A, S(F(Z)))\big) \le \varepsilon,$$

where S uses fresh independent randomness. The analogous statement with computational indistinguishability holds for a computationally private construction when the function descriptions and the bit lengths of Z, A are polynomially bounded in the security parameter.

For q adaptive uses, suppose that each input is chosen from the current state and a fresh uniform encoding tape is then sampled independently of the complete state after this choice, including the chosen input. If the experiment accesses each tape only through its encoding output, the statistical errors add to at most $q\varepsilon$; the computational analogue holds for polynomially many uses in an efficient experiment.

Representations and size. We represent $f : \{0,1\}^N \to \{0,1\}^b$ by one mod-2 branching program per output bit, following [AIK06b]. Each program is a directed acyclic graph without parallel edges, with designated initial and terminal vertices and edges labelled by input literals or 1. On input x, it outputs the parity of the number of paths from the initial vertex to the terminal vertex whose labels all evaluate to one. The total representation size σ is the sum of the numbers of vertices in these programs plus $N + b$. Circuit representations are Boolean circuits of fan-in at most two, with total size equal to the number of non-input gates plus $N + b$. For multiparty functionalities, N and b are the total input and output lengths, and the representation also specifies the assignment of these bits to the parties.

Fact 3.4 (Degree-three randomized encodings [IK00, IK02, AIK06b])**.** *Let $g : \{0,1\}^N \to \{0,1\}$ have a mod-2 branching-program representation of total size σ. Then g has a perfectly correct, perfectly private RE with randomness and output lengths $O(\sigma^2)$, whose coordinate polynomials are sums of monomials containing at most one input variable and at most two randomness variables. An explicit polynomial description of the encoder, together with a decoder and simulator, can be constructed in time polynomial in σ, and encoding, decoding and simulation also take time polynomial in σ.*

3.3 Multiparty randomized encodings

Definition 3.5 (Multiparty randomized encodings [ABT18, AIKP22])**.** Let $f = (f_1, \ldots, f_n) : \prod_{i=1}^n X_i \to \prod_{i=1}^n Y_i$ be a deterministic n-party functionality with finite nonempty input and output spaces, where party i holds x_i and is prescribed $f_i(\mathbf{x})$. Write $f_T(\mathbf{x}) = (f_i(\mathbf{x}))_{i \in T}$. Let $R_0, \ldots, R_n, Z_1, \ldots, Z_n$ be finite nonempty sets, and consider an n-party randomized functionality

$$\hat{f} : \left(\prod_{i=1}^n (X_i \times R_i)\right) \times R_0 \longrightarrow \prod_{i=1}^n Z_i.$$

For an input $\mathbf{x} = (x_1, \ldots, x_n)$, independently sample $r_i \leftarrow R_i$ for $i = 0, \ldots, n$, write $\mathbf{r} = (r_1, \ldots, r_n)$, and denote the resulting vector of party outputs by $\hat{\mathbf{y}} = \hat{f}((x_1, r_1), \ldots, (x_n, r_n); r_0)$. The *view* of a coalition $T \subseteq [n]$ is

$$V_T(\mathbf{x}) = \big(\mathbf{x}[T], \mathbf{r}[T], \hat{\mathbf{y}}[T]\big),$$

including its inputs, complete local randomness and encoded outputs, but excluding the internal randomness r_0.

For $\delta, \varepsilon \in [0, 1]$ and $\tau \in \{0, \ldots, n\}$, the functionality $\hat{f}$ is a *δ-correct, (τ, ε)-private multiparty randomized encoding* (MPRE) of f if the following conditions hold:

- *δ-correctness.* There is a deterministic decoder Dec such that, for every input $\mathbf{x}$ and every $i \in [n]$,
$$\Pr_{r_0, \mathbf{r}}\big[\mathsf{Dec}(i, \hat{y}_i, x_i, r_i) \neq f_i(\mathbf{x})\big] \leq \delta.$$
- *(τ, ε)-privacy.* There is a randomized simulator Sim such that, for every input $\mathbf{x}$ and every coalition $T \subseteq [n]$ with $|T| \leq \tau$,
$$\mathsf{Sim}\big(T, \mathbf{x}[T], f_T(\mathbf{x})\big) \approx_\varepsilon V_T(\mathbf{x}).$$

The encoding is *perfectly correct* when $\delta = 0$ and *perfectly τ-private* when $\varepsilon = 0$. We call privacy *full* when it holds against all coalitions, that is, when $\tau = n$. The encoding has *no internal randomness* if R_0 contains only one element.

Definition 3.6 (Effective degree and 2MPRE [AIKP22])**.** A randomized n-party functionality

$$g : \left(\prod_{i=1}^{n} W_i\right) \times \mathbb{F}^\rho \longrightarrow \prod_{i=1}^{n} \mathbb{F}^{\ell_i}$$

has *effective degree at most d* over $\mathbb{F}$ if there are local preprocessing maps $h_i : W_i \to \mathbb{F}^{a_i}$ and a polynomial map $p : (\prod_{i=1}^n \mathbb{F}^{a_i}) \times \mathbb{F}^\rho \to \prod_{i=1}^n \mathbb{F}^{\ell_i}$ such that

$$g(\mathbf{w}; \mathbf{r}_0) = p\big(h_1(w_1), \ldots, h_n(w_n); \mathbf{r}_0\big)$$

for every $\mathbf{w} = (w_1, \ldots, w_n)$ and $\mathbf{r}_0 \in \mathbb{F}^\rho$, and p has total degree at most d jointly in the preprocessed coordinates and the internal randomness. The local maps h_i have no degree restriction.

An MPRE is a *quadratic MPRE*, or *2MPRE*, over $\mathbb{F}$ if $R_0 = \mathbb{F}^\rho$, $Z_i = \mathbb{F}^{\ell_i}$, and its effective degree is at most two, with local inputs $w_i = (x_i, r_i)$. The vectors $h_i(x_i, r_i)$ are its *local values*. All dimensions may be zero.

Definition 3.7 (Efficiency and asymptotic security)**.** Following [ABT18], an encoding construction takes a representation of a functionality f and 1^λ and produces an encoder, a decoder and a simulator. The representation of f and 1^λ are public parameters, omitted from the notation. We use the following conventions.

- *Efficiency.* For an n-party functionality of representation size σ, the construction is efficient if its algorithms can be uniformly constructed and evaluated in time polynomial in $n + \sigma + \lambda$, including the local preprocessing maps and the public polynomial map when present. Random tapes consist of independent uniform bits and have polynomially bounded length. We measure *encoding complexity* by the total bit length of the encoding randomness, local values when present, and encoded outputs.

- *Statistical security.* The correctness and statistical privacy bounds $\delta(\lambda)$ and $\varepsilon(\lambda)$ hold uniformly over all represented functionalities, valid inputs and allowed coalitions. We call the construction statistical when both bounds are negligible in λ.
- *Computational privacy.* An efficient construction is computationally private if the real and simulated views in Definition 3.5 are computationally indistinguishable for every sequence of represented functionalities, valid inputs and allowed coalitions, provided the numbers of parties and representation sizes are polynomially bounded in λ. The simulator is the one supplied by the construction.

The same conventions apply to ordinary REs using the correctness and privacy conditions of Definition 3.1, with party counts, coalitions and local values omitted.

4 Statistical Randomized Encodings of Degree Two

We now give the full construction and analysis of the quadratic encoding of $g^\star(a,b,c,d) = abc + d$ described in Section 2. After proving its correctness and privacy, we use it to obtain encodings of arbitrary Boolean functions and the computational variant for circuits.

4.1 Our encoding

Our scheme is based on the recursive sharing schemes described in Section 2 and it has two public parameters: a sharing depth $j \geq 0$ and a number $h \geq 3$ of rows in each encoded matrix. Both are integers, and we write $L = 4^j$ for the number of leaves. We recall that we share the input bits a and c using two linear secret-sharing schemes over $\mathbb{F}_2$, whose sharing procedures are $\mathsf{Sh}_A(a;\theta) = (a+\theta, a+\theta, \theta, \theta)$ and $\mathsf{Sh}_C(c;\theta_1,\theta_2) = (c+\theta_1, \theta_1, c+\theta_2, \theta_2)$, where $\theta, \theta_1, \theta_2$ are independent uniform bits. The four coordinates are indexed by $[4] = \{1,2,3,4\}$, and the respective access structures are

$$\Gamma_A = \{S \subseteq [4] : S \cap \{1,2\} \neq \varnothing \text{ and } S \cap \{3,4\} \neq \varnothing\},$$
$$\Gamma_C = \{S \subseteq [4] : \{1,2\} \subseteq S \text{ or } \{3,4\} \subseteq S\}.$$

Applying both schemes recursively defines the two sharing procedures Sh_A^j and Sh_C^j whose output coordinates correspond to the leaves of complete four-ary trees of depth j, with

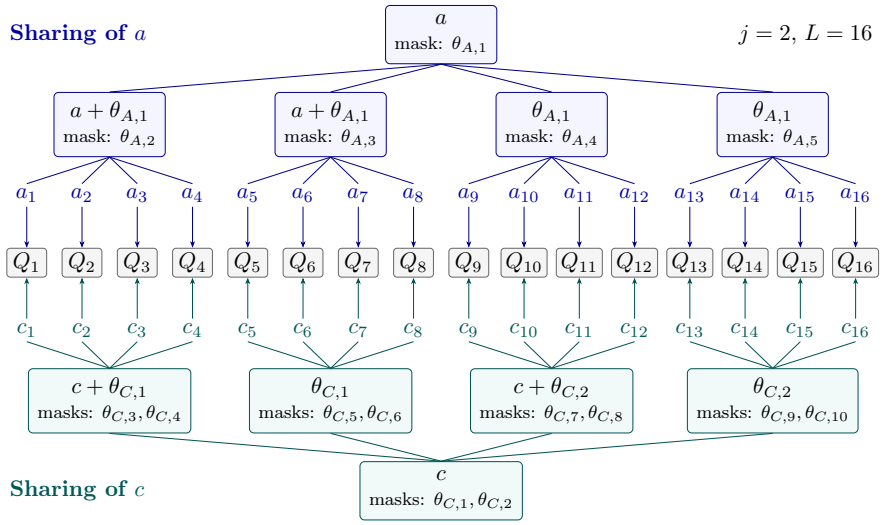
$$\sum_{i=1}^{L} a_i c_i = ac. \tag{1}$$

Together with an additive sharing of d, this gives leaf bits $y_i = a_i b c_i + d_i$ whose parity is $abc + d$. We apply the matrix-rank construction of Ishai and Kushilevitz [IK00, IK02] independently at each leaf and decode by taking the parity of the rank-test results. Figure 3 illustrates the construction, and Construction 4.1 specifies the encoder and decoder.

Construction 4.1 (The encoding $E_{j,h}$)

PUBLIC PARAMETERS. Integers $j \geq 0$ and $h \geq 3$; set $L = 4^j$ and $m = (L-1)/3$.
INPUT. Four bits $(a,b,c,d) \in \mathbb{F}_2^4$.

- **Randomness sampling and sharing.** The encoder proceeds as follows.



$$Q_i = N_i M(a_i, b, c_i, d_i), \qquad \sum_{i=1}^{L} d_i = d, \qquad N_i \leftarrow \mathbb{F}_2^{h\times 3} \text{ independently}$$
$$E_{j,h}(a, b, c, d; R) = \mathrm{vec}(Q_1) \,\|\, \cdots \,\|\, \mathrm{vec}(Q_L) \in \mathbb{F}_2^{3hL}$$
$$\mathrm{Dec}_{j,h}(Q_1, \ldots, Q_L) = \textstyle\sum_{i=1}^{L} \mathbf{1}\{\mathrm{rank}(Q_i) = 3\} \quad \text{in } \mathbb{F}_2$$

Figure 3: The two recursive sharings and their use in the encoding, at depth $j = 2$. Corresponding leaf shares a_i, c_i are combined with b, d_i, N_i to form the encoded matrix Q_i.

1. Sample independent and uniform random values $R = (\boldsymbol{\theta}_A, \boldsymbol{\theta}_C, \bar{\mathbf{d}}, N_1, \ldots, N_L)$, where $\boldsymbol{\theta}_A \in \mathbb{F}_2^m$, $\boldsymbol{\theta}_C \in \mathbb{F}_2^{2m}$, $\bar{\mathbf{d}} \in \mathbb{F}_2^{L-1}$ and $N_i \in \mathbb{F}_2^{h\times 3}$.
2. Set $\mathbf{a} = (a_1, \ldots, a_L) = \mathsf{Sh}_A^j(a; \boldsymbol{\theta}_A)$ and $\mathbf{c} = (c_1, \ldots, c_L) = \mathsf{Sh}_C^j(c; \boldsymbol{\theta}_C)$. Generate the additive sharing by setting $d_i = \bar{d}_i$ for $i < L$ and $d_L = d + \sum_{\ell=1}^{L-1} \bar{d}_\ell$. The same input bit b is used at every leaf. For $j = 0$, both mask vectors and the prefix are empty, and $(a_1, c_1, d_1) = (a, c, d)$.

- **Encoding.** For every $i \in [L]$, compute

$$Q_i = N_i \begin{pmatrix} 1 & 0 & c_i \\ a_i & 1 & 0 \\ 0 & b & d_i \end{pmatrix}.$$

  Output $E_{j,h}(a, b, c, d; R) = (Q_1, \ldots, Q_L)$.

- **Decoding.** On input $(Q_1, \ldots, Q_L)$ of $h \times 3$ binary matrices, output $\mathsf{Dec}_{j,h}(Q_1, \ldots, Q_L) = \sum_{i=1}^{L} \mathbf{1}\{\mathrm{rank}(Q_i) = 3\}$ in $\mathbb{F}_2$.

4.2 Complexity, correctness and privacy

We can write the L output matrices as a string of $3hL$ bits by listing their entries in leaf, row, and column order. The random tape consists of $3m = L - 1$ masks for the two recursive sharings, $L - 1$ bits for the additive sharing of d, and $3hL$ bits for the coefficient matrices. Thus the encoder has the form $E_{j,h} : \mathbb{F}_2^4 \times \mathbb{F}_2^\rho \longrightarrow \mathbb{F}_2^{3hL}$, with $\rho = 2(L - 1) + 3hL$.

We establish the degree, complexity, correctness and privacy bounds for arbitrary integers $j \geq 0$

and $h \geq 3$, then choose both parameters to obtain exponentially small errors.

4.2.1 Degree and complexity

Proposition 4.2 (Degree and complexity of $E_{j,h}$)**.** *For every $j \geq 0$ and $h \geq 3$, with $L = 4^j$, the encoding map $E_{j,h}$ has total degree at most two jointly in the original input bits and the independent uniform bits of its random tape. The random tape has length $2(L-1)+3hL$, the encoded output has length $3hL$, and the encoder and decoder use $O(Lh)$ operations over $\mathbb{F}_2$. Writing each output bit explicitly as a sum of monomials in the original input bits and the independent bits of the random tape gives $O(Lh(j+1))$ monomials in total. These expressions can be constructed in time polynomial in L and h.*

Proof. For the degree, if (n_1, n_2, n_3) is a row of N_i, its encoded row is

$$(n_1, n_2, n_3) \cdot M(a_i, b, c_i, d_i) = (n_1 + n_2 a_i,\ n_2 + n_3 b,\ n_1 c_i + n_3 d_i). \tag{2}$$

Each n_ℓ is an independent uniformly random coordinate, and each of a_i, c_i, d_i is affine in the original input and random coordinates. Every entry in (2) therefore has total degree at most two in those original variables.

The random tape consists of the $m + 2m = L - 1$ masks of the two recursive sharings, the $L - 1$ bits of the additive prefix and the $3hL$ entries of the coefficient matrices, so its length is $\rho = 2(L-1) + 3hL$, while the encoding consists of L matrices of size $h \times 3$, that is, of $3hL$ bits. The encoder performs a constant number of operations at each internal node of the sharing trees and for each additive share, and $O(h)$ operations for each matrix Q_i, whereas the decoder performs L Gaussian eliminations on $h \times 3$ matrices, each of which costs $O(h)$ operations because the number of columns is fixed; both therefore use $O(Lh)$ operations. Finally, every leaf share a_i or c_i is a sum of at most $j+1$ variables, since it contains at most the input bit and one mask from each level of the recursion, and every additive share is a single variable except d_L, which has L terms; by (2), every entry of the encoding therefore expands into $O(j+1)$ monomials, except the h entries of Q_L that involve d_L, which contribute $O(L)$ monomials each, for a total of $O(Lh(j+1))$ monomial occurrences, and this expansion can be computed in time polynomial in L and h. □

4.2.2 Correctness

Proposition 4.3 (Correctness of $E_{j,h}$)**.** *For every pair of integers $j \geq 0$ and $h \geq 3$, let $L = 4^j$. For every $(a, b, c, d) \in \mathbb{F}_2^4$, the decoder of $E_{j,h}$ satisfies*

$$\Pr_R\big[\mathsf{Dec}_{j,h}(E_{j,h}(a, b, c, d; R)) \neq abc + d\big] \leq \begin{cases} \frac{7}{2}\, L\, 2^{-h}, & j \geq 1, \\ 7 \cdot 2^{-h}, & j = 0, \end{cases}$$

where the probability is over the uniformly sampled random tape R.

Proof. We first prove Equation (1) by induction on j. At depth zero it is immediate

$$\begin{aligned} \big\langle \mathsf{Sh}_A(a;\theta), \mathsf{Sh}_C(c;\theta_1,\theta_2) \big\rangle &= (a+\theta)(c+\theta_1) + (a+\theta)\theta_1 + \theta(c+\theta_2) + \theta\theta_2 \\ &= (a+\theta)c + \theta c = ac. \end{aligned}$$

At depth $j+1$, we first apply the one-level sharing rules to a and c, obtaining

$$(\alpha_1, \ldots, \alpha_4) = (a+\theta, a+\theta, \theta, \theta), \qquad (\gamma_1, \ldots, \gamma_4) = (c+\theta_1, \theta_1, c+\theta_2, \theta_2),$$

where $\theta, \theta_1, \theta_2$ are the masks used at the two roots. For each $\nu \in [4]$, let $\mathbf{a}^{(\nu)}$ and $\mathbf{c}^{(\nu)}$ be the leaf vectors obtained by sharing α_ν and γ_ν, respectively, for another j levels. The full vectors $\mathbf{a}$ and $\mathbf{c}$ concatenate these four pairs of leaf vectors in the same order. Applying the induction hypothesis to each pair and then the one-level identity gives

$$\langle \mathbf{a}, \mathbf{c} \rangle = \sum_{\nu=1}^{4} \langle \mathbf{a}^{(\nu)}, \mathbf{c}^{(\nu)} \rangle = \sum_{\nu=1}^{4} \alpha_\nu \gamma_\nu = ac.$$

This holds for every choice of the sharing masks. Together with $\sum_i d_i = d$, it gives

$$\textstyle\sum_{i=1}^{L} y_i = b \sum_{i=1}^{L} a_i c_i + \sum_{i=1}^{L} d_i = abc + d. \tag{3}$$

Thus, if every rank test recovers its leaf bit correctly, the parity returned by the decoder is the correct output.

Having shown that the leaf bits sum to the prescribed output, we now bound the probability that the rank tests give an incorrect final answer. We first analyze a single rank test when its leaf inputs are fixed, and then average over the randomness used to generate those inputs.

For the single-leaf analysis, we additionally condition on the sharing randomness $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_C, \bar{\mathbf{d}})$, which determines all the shares a_i, c_i, d_i and hence all the leaf bits y_i. The coefficient matrices remain independently and uniformly distributed, because they were sampled independently of the sharing randomness.

At leaf i, the decoder receives $Q_i = N_i M_i$, where $M_i = M(a_i, b, c_i, d_i)$, and returns one precisely when Q_i has rank three. Since $\det M_i = a_i b c_i + d_i = y_i$, there are two cases. If $y_i = 0$, then M_i is singular, so $\operatorname{rank}(Q_i) = \operatorname{rank}(N_i M_i) \le \operatorname{rank}(M_i) \le 2$. The rank test therefore returns zero for every choice of N_i, and this leaf is always decoded correctly. If $y_i = 1$, then M_i is invertible, so $\operatorname{rank}(Q_i) = \operatorname{rank}(N_i)$. In this case, the rank test is incorrect precisely when the random coefficient matrix N_i has rank less than three. We therefore bound the probability of this event. Let $\delta_h = \Pr_{N \leftarrow \mathbb{F}_2^{h \times 3}}[\operatorname{rank}(N) < 3]$, an $h \times 3$ matrix has rank less than three precisely when its kernel contains a nonzero vector, so this event occurs if $N\mathbf{z} = 0$ for at least one nonzero $\mathbf{z} \in \mathbb{F}_2^3$. For any fixed nonzero $\mathbf{z}$, the inner product of a uniform row of N with $\mathbf{z}$ is a uniform bit, because at least one coordinate of $\mathbf{z}$ is one. The h rows are independent, and hence $\Pr[N\mathbf{z} = 0] = 2^{-h}$. There are $2^3 - 1 = 7$ nonzero vectors to consider, so the union bound gives $\delta_h \le \sum_{\mathbf{z} \neq 0} \Pr[N\mathbf{z} = 0] = 7 \cdot 2^{-h}$. By the law of total probability, $\Pr_R[\text{leaf } i \text{ is decoded incorrectly}] = \Pr_R[y_i = 1]\, \delta_h$. To evaluate the remaining probability, suppose $L \ge 2$. Each additive share d_i is a uniform bit: the first $L - 1$ coordinates are sampled uniformly, and the last coordinate is the fixed bit d plus the sum of a nonempty collection of independent uniform prefix bits. Moreover, the entire additive sharing is sampled independently of the recursive masks. Thus, even after conditioning on a_i and c_i, the bit d_i remains uniform, and adding the fixed bit $a_i b c_i$ preserves this uniform distribution. Hence we have that $\Pr_R[y_i = 1] = \frac{1}{2}$ and $\Pr_R[\text{leaf } i \text{ is decoded incorrectly}] = \frac{\delta_h}{2}$. Equation (3) shows that recovering every leaf bit correctly guarantees that their parity is the prescribed output. An incorrect final answer therefore requires at least one incorrect leaf answer, and another union bound gives

$$\Pr_R\big[\mathsf{Dec}_{j,h}(E_{j,h}(a, b, c, d; R)) \neq abc + d\big] \le \textstyle\sum_{i=1}^{L} \Pr_R[\text{leaf } i \text{ is decoded incorrectly}] = \frac{L\delta_h}{2} \le \frac{7}{2} L\, 2^{-h}. \tag{4}$$

When $L = 1$, the only leaf bit equals $abc + d$ and is determined by the original input. The rank test is always correct when this bit is zero and has error probability δ_h when it is one, giving correctness error at most $\delta_h \le 7\, 2^{-h}$. □

Once the sharing depth is fixed, increasing h makes the correctness error arbitrarily small. We choose both parameters after completing the privacy analysis.

4.2.3 Privacy

Proposition 4.4 (Privacy of $E_{j,h}$)**.** *Let $j \geq 4$ and $h \geq 3$ be integers, and set $L = 4^j$. There exists a randomized simulator $\mathsf{Sim}_{j,h}$, using $O(Lh)$ operations, such that for every $(a, b, c, d) \in \mathbb{F}_2^4$,*

$$\mathrm{SD}\big(E_{j,h}(a, b, c, d; R), \mathsf{Sim}_{j,h}(abc + d)\big) \leq 5^{-2^{j-4}} + \tfrac{1}{2}\, 20^{-2^{j-4}} \leq \tfrac{3}{2}\, 5^{-2^{j-4}},$$

where R is a uniformly sampled random tape.

Proof. Given y, the simulator samples a fresh uniform tape R' and outputs $\mathsf{Sim}_{j,h}(y) = E_{j,h}(0, 0, 0, y; R')$, using $O(Lh)$ operations. Fix an input (a, b, c, d) with $abc + d = y$. We compare its encoding with this simulation by replacing the sharing roots a and c by zero, one at a time, while keeping the distribution of the leaf bits fixed. The argument first gives a bound for every $j \geq 0$, which we then specialize to $j \geq 4$.

At a high level, in Step 1, we give an equivalent way to sample the encoding from the leaf bits and selected shares of a and c, which identifies the share values on which the matrix distributions depend. Step 2 shows that any fixed set of shares either reconstructs its root or has a distribution independent of that root. These two facts prepare the hybrid argument in Step 3: we introduce five hybrids connecting the real encoding to the simulator, using Step 1 for the two exact transitions and Step 2 to bound the effect of replacing a and then c by zero. This reduces the privacy error to two reconstruction probabilities. Finally, Step 4 bounds those probabilities through the recursive structure of the sharings and concludes the proof.

Step 1: Identifying the information needed to sample the encoding. We begin by describing another way to generate the encoded matrices, with exactly the same distribution as in the original construction. This description will show which shares are sufficient to sample the encoding, so that the remaining proof can focus on the information carried by those shares.

First, we fix the sharing randomness $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_C, \bar{\mathbf{d}})$. This fixes all the shares a_i, c_i, d_i and hence the leaf bits $y_i = a_i b c_i + d_i$, while the coefficient matrices N_i remain independent and uniform. Each row of $Q_i = N_i M_i$, where $M_i = M(a_i, b, c_i, d_i)$, is therefore a uniform sample from the row space of M_i, independently of all other rows. If $y_i = 1$, then M_i is invertible. If $y_i = 0$, its first two columns are linearly independent and $M_i(c_i, a_i c_i, 1)^{\mathsf{T}} = 0$, so its row space is the plane orthogonal to $(c_i, a_i c_i, 1)$. Thus

$$\mathrm{Row}(M_i) = \begin{cases} \mathbb{F}_2^3, & y_i = 1, \\ \{(z_1, z_2, z_3) \in \mathbb{F}_2^3 : z_3 = c_i z_1 + a_i c_i z_2\}, & y_i = 0. \end{cases} \tag{5}$$

We can sample directly from these spaces without forming N_i or M_i. To do so, we define a randomized procedure $\mathsf{RowSamp}(y_i, c_i, a_i)$ that samples three fresh independent uniform bits U, V, T and returns

$$\mathsf{RowSamp}(y_i, c_i, a_i) = \begin{cases} (U, V, T), & y_i = 1, \\ (U, V, 0), & y_i = 0,\ c_i = 0, \\ (U, V, U + a_i V), & y_i = 0,\ c_i = 1. \end{cases} \tag{6}$$

When $y_i = 1$, the returned row is uniform in $\mathbb{F}_2^3$; when $y_i = 0$, the first two coordinates U, V determine the third through the equation of the plane, so choosing them uniformly gives a uniform point in that plane. Thus $\mathsf{RowSamp}$ has exactly the same row distribution as the original construction. Running it h times at each leaf, with fresh independent bits for every row and every leaf, reproduces the joint distribution of all the encoded matrices for the fixed sharing values.

This procedure also makes clear which shares are needed. If $y_i = 1$, the procedure uses no share values. If $y_i = 0$, it first needs c_i to determine which plane to use, and it needs a_i only when $c_i = 1$.

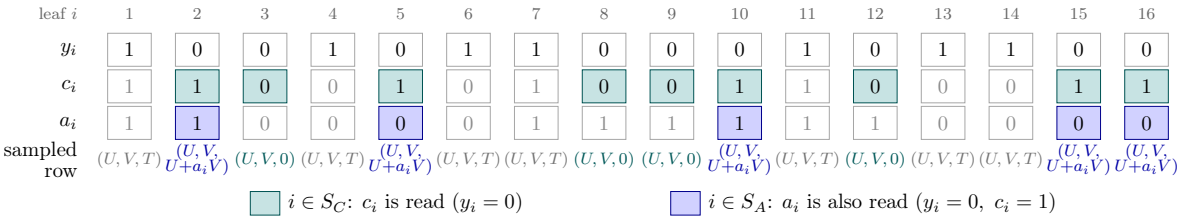


Figure 4: The shares used by the row sampler, illustrated for $L = 16$. First, the zero entries of $\mathbf{y}$ select the c-shares to be read. Among these leaves, those with $c_i = 1$ also require the corresponding a-share. The shaded entries mark the selected shares; together with $\mathbf{y}$, they are sufficient to sample all the encoded matrices. Every sampled row at every leaf uses fresh independent random bits.

Accordingly, we define

$$S_C(\mathbf{y}) = \{i \in [L] : y_i = 0\}, \qquad S_A(\mathbf{c}, \mathbf{y}) = \{i \in [L] : y_i = 0,\ c_i = 1\},$$

where $\mathbf{y} = (y_1, \ldots, y_L)$ to identify the index of useful shares and call $\mathbf{c}[S_C]$ and $\mathbf{a}[S_A]$ the *selected shares*. The selection takes place in two stages: $\mathbf{y}$ tells us which c-shares to read, and those c-shares tell us which a-shares to read. This means that $\mathbf{y}$, $\mathbf{c}[S_C]$ and $\mathbf{a}[S_A]$ together contain all the information needed to run the row sampler at every leaf. We list each selected share vector in increasing order of its leaf indices. Figure 4 illustrates this sampling rule. We have seen how to generate the matrices once the leaf bits and selected shares are known, so we now turn to the leaf bits themselves. We will show that we can sample them using only the prescribed output y, independently of the two recursive sharings.

Suppose that we have already sampled the recursive masks $(\boldsymbol{\theta}_A, \boldsymbol{\theta}_C)$, which determine the share vectors $\mathbf{a}, \mathbf{c}$, but have not yet sampled the additive sharing of d. The vector $\mathbf{d}$ is then uniform among all vectors whose coordinates sum to d. To obtain the leaf bits, we add $(a_i b c_i)_{i \in [L]}$ to this vector, where the values being added have already been determined by the recursive masks and the input. Since $\sum_i a_i b c_i = abc$, this addition maps the vectors with coordinate sum d bijectively onto the vectors with coordinate sum $abc + d = y$. Let $\mathcal{P}_y = \left\{ \boldsymbol{\eta} \in \mathbb{F}_2^L : \sum_{i=1}^L \eta_i = y \right\}$, we obtain, for every choice of the recursive masks,

$$\mathbf{y} \;\big|\; (\boldsymbol{\theta}_A, \boldsymbol{\theta}_C) \;\equiv\; U_{\mathcal{P}_y}. \tag{7}$$

Since we obtain the same uniform distribution for every choice of these masks, $\mathbf{y}$ is independent of both mask vectors jointly.

We can now generate the same encoding distribution by sampling $\mathbf{y}$ uniformly from $\mathcal{P}_y$, independently generating the recursive sharings of a and c, and applying $\mathsf{RowSamp}$ with the selected shares. The advantage of this sampling procedure is that, once we know the prescribed output y, we no longer need b or d, and the only remaining dependence on the input is through the selected shares of a and c. We are therefore left to show that replacing these two sharing roots by zero, while keeping y unchanged, changes the resulting distribution by only a small amount.

Step 2: Reconstruction and privacy of the selected shares. We have seen that, once the leaf bits are known, we only need the selected shares of a and c to generate the encoded matrices. To understand when we can replace either root by zero without changing the distribution of the selected shares, we first identify the sets of leaves whose shares allow us to reconstruct that root.

Definition 4.5 (Authorized sets)**.** For $X \in \{A, C\}$, we call a set $S \subseteq [L]$ *authorized* for Sh_X^j if there exists a map $\mathrm{Rec}_S : \mathbb{F}_2^{|S|} \to \mathbb{F}_2$ satisfying $\mathrm{Rec}_S\big(\mathsf{Sh}_X^j(s; \boldsymbol{\theta})[S]\big) = s$ for every root bit s and every mask

vector $\boldsymbol{\theta}$. We write $\mathrm{Auth}^j_X(S) = 1$ when this condition holds; otherwise, we call S *unauthorized* and write $\mathrm{Auth}^j_X(S) = 0$.

The definition tells us whether a set of shares suffices to reconstruct the root, but it leaves open whether an unauthorized set might still reveal partial information about it. We now show that, for both recursive sharings Sh^j_A and Sh^j_C, the shares in any unauthorized set have exactly the same joint distribution for root values zero and one.

Lemma 4.6 (Information in a set of leaves)**.** *For each $X \in \{A, C\}$ and each set $S \subseteq [L]$, if S is authorized for Sh^j_X, then it admits a linear reconstruction map. If S is unauthorized, then*

$$\mathsf{Sh}^j_X(0; \boldsymbol{\theta})[S] \equiv \mathsf{Sh}^j_X(1; \boldsymbol{\theta})[S],$$

where both distributions are taken over uniformly sampled sharing masks $\boldsymbol{\theta}$.

Proof. We consider either recursive sharing Sh^j_X and an arbitrary set of leaves $S \subseteq [L]$. Each one-level share is either a mask or the sum of the bit being shared and a mask. Applying these rules recursively therefore makes every leaf share linear jointly in the root and all sharing masks. For each leaf i, linearity lets us write its share as $g_i s + \sum_{t=1}^{m_X} \Lambda_{i,t}\theta_t$, where the coefficients $g_i, \Lambda_{i,t} \in \mathbb{F}_2$ are fixed by the sharing rules. Here $m_A = m$ and $m_C = 2m$ are the numbers of masks used by the two sharings. Collecting the coefficients of the root in a vector $\mathbf{g} \in \mathbb{F}_2^L$ and those of the masks in a matrix $\Lambda \in \mathbb{F}_2^{L \times m_X}$ gives

$$\mathsf{Sh}^j_X(s; \boldsymbol{\theta}) = s\mathbf{g} + \Lambda\boldsymbol{\theta}.$$

Each row of Λ describes the mask contributions to one leaf share. Keeping only the shares indexed by S therefore gives

$$\mathsf{Sh}^j_X(s; \boldsymbol{\theta})[S] = s\mathbf{g}[S] + \Lambda_S\boldsymbol{\theta},$$

where $\mathbf{g}[S]$ contains the corresponding root coefficients and Λ_S consists of the rows of Λ indexed by S.

To understand the distribution of these shares, we consider the subspace $V = \mathrm{Im}(\Lambda_S)$ of all possible masking vectors on S. Uniformly sampled masks make $\Lambda_S\boldsymbol{\theta}$ uniform on V, even if several mask vectors produce the same value. Indeed, for any $v \in V$, once we have one solution $\boldsymbol{\theta}_0$ to $\Lambda_S\boldsymbol{\theta} = v$, all solutions are exactly the vectors in $\boldsymbol{\theta}_0 + \ker(\Lambda_S)$. Every value in V therefore has the same number of preimages. It follows that the selected shares are uniform on V when the root is zero and uniform on $\mathbf{g}[S] + V$ when the root is one. We now distinguish two cases.

Suppose first that $\mathbf{g}[S] \in V$. Adding $\mathbf{g}[S]$ to V leaves the subspace unchanged, so the two root values give exactly the same joint share distribution. In particular, every possible share vector can arise from either root value, and no reconstruction map can always recover the root. Thus S is unauthorized, and its shares reveal no information about the root.

Suppose instead that $\mathbf{g}[S] \notin V$. We will construct a linear map that removes the contribution of the masks while recovering the root s from the selected shares. To do so, we take a basis of V, adjoin $\mathbf{g}[S]$, and extend the resulting linearly independent list to a basis of $\mathbb{F}_2^{|S|}$. Then, we define a linear map $\psi : \mathbb{F}_2^{|S|} \to \mathbb{F}_2$ by assigning value one to the basis vector $\mathbf{g}[S]$ and value zero to every other basis vector. By construction, ψ vanishes on V and satisfies $\psi(\mathbf{g}[S]) = 1$. Applying it to the selected shares therefore gives

$$\psi\big(s\mathbf{g}[S] + \Lambda_S\boldsymbol{\theta}\big) = s\psi(\mathbf{g}[S]) + \psi(\Lambda_S\boldsymbol{\theta}) = s$$

for every root bit s and every mask vector $\boldsymbol{\theta}$. Thus S is authorized, with linear reconstruction map ψ. Every set S falls into one of these two cases. An authorized set must fall into the second case

and therefore admits a linear reconstruction map, while an unauthorized set must fall into the first case and therefore has identical share distributions for the two root values. □

We now bound the probability that the shares selected by $\mathsf{RowSamp}$ form an authorized set, which will allow us to control the effect of replacing each root by zero.

Step 3: Bounding the privacy error. We will compare the selected-share distributions when we change the roots of the two sharings. By Lemma 4.6, changing the root leaves the distribution of any fixed unauthorized set of shares unchanged. We therefore need to bound the probability that the selected shares form an authorized set. Since the row sampler uses the c-shares at zero leaves and the a-shares only at zero leaves whose c-shares are one, we need to bound the reconstruction probability for each of these two sets.

For the recursive analysis, we first work with independent uniform leaf bits, without the encoder's parity constraint, so that the leaf bits in different child subtrees are independent. We will then obtain bounds for the encoder's actual leaf bits by conditioning on their prescribed parity. Let $\tilde{\mathbf{y}} = (\tilde{y}_1, \dots, \tilde{y}_L)$ be uniform on $\mathbb{F}_2^L$ and, for a fixed root bit c, independently generate $\mathbf{c} = \mathsf{Sh}_C^j(c; \boldsymbol{\theta}_C)$ using uniform masks. For $X \in \{A, C\}$, let $\mathrm{Auth}_X^j(S)$ equal one if S is authorized for Sh_X^j, and zero otherwise. We define

$$\begin{aligned} p_j &= \Pr[\mathrm{Auth}_C^j(\{i : \tilde{y}_i = 0\}) = 1], \\ q_j(c) &= \Pr[\mathrm{Auth}_A^j(\{i : \tilde{y}_i = 0,\ c_i = 1\}) = 1], \qquad q_j^* = \max\{q_j(0), q_j(1)\}. \end{aligned} \tag{8}$$

Thus p_j is the probability that the selected c-shares, namely those with $\tilde{y}_i = 0$, form an authorized set in the C-sharing scheme. The selection depends only on $\tilde{\mathbf{y}}$, and whether the set is authorized does not depend on the root value, so p_j is independent of c. Similarly, $q_j(c)$ is the probability that the a-shares selected by $\tilde{y}_i = 0$ and $c_i = 1$ form an authorized set in the A-sharing scheme. Here the selection also depends on the sampled c-shares, so the probability is taken over both $\tilde{\mathbf{y}}$ and the masks used to share c. It may therefore depend on c, whereas a affects neither the selection nor whether the selected set is authorized. Taking q_j^* gives a bound valid for both values of c.

The encoder's leaf bits must sum to the output y, so we now condition the independent bits $\tilde{y}_1, \dots, \tilde{y}_L$ on $\sum_i \tilde{y}_i = y$. By Equation (7), the distribution of $\mathbf{y}$ is exactly that of $\tilde{\mathbf{y}}$ conditioned on $\sum_i \tilde{y}_i = y$. Since this condition involves only $\tilde{\mathbf{y}}$, the sharing masks remain independent and uniform. Let $\mathcal{E}$ denote either of the reconstruction events defining p_j and $q_j(c)$ in Equation (8). A uniform vector has either parity with probability $1/2$, so $\Pr\left[\mathcal{E} \middle| \sum_i \tilde{y}_i = y\right] = \Pr\left[\mathcal{E} \text{ and } \sum_i \tilde{y}_i = y\right] / \Pr\left[\sum_i \tilde{y}_i = y\right] = 2\Pr\left[\mathcal{E} \text{ and } \sum_i \tilde{y}_i = y\right] \le 2\Pr[\mathcal{E}]$. Thus imposing the required parity can increase either reconstruction probability by at most a factor of two, giving

$$\Pr[\mathrm{Auth}_C^j(S_C(\mathbf{y})) = 1] \le 2p_j, \text{ and } \Pr[\mathrm{Auth}_A^j(S_A(\mathbf{c}, \mathbf{y})) = 1] \le 2q_j(c) \le 2q_j^*. \tag{9}$$

The next lemma uses these bounds to compare the real encoding with its simulation through a sequence of hybrids. We will then bound p_j and q_j^* as functions of the sharing depth.

Lemma 4.7 (Simulation from selected shares)**.** *For every $j \ge 0$ and $h \ge 3$, the encoding $E_{j,h}$ has simulator $\mathsf{Sim}_{j,h}$ and privacy error at most $\varepsilon_{j,h} = \min\{1,\ 2q_j^* + 2p_j\}$.*

Proof. The proof is by a sequence of hybrids, summarized in Figure 5. We view each hybrid output as a list of L matrices in $\mathbb{F}_2^{h\times 3}$, as in the construction. We use the row sampler from Step 1, keeping the prescribed output $y = abc + d$ fixed and the leaf vector uniform on $\mathcal{P}_y$ throughout. The first and last transitions preserve the distribution exactly, while the two middle transitions replace the sharing roots a and c by zero, in that order.

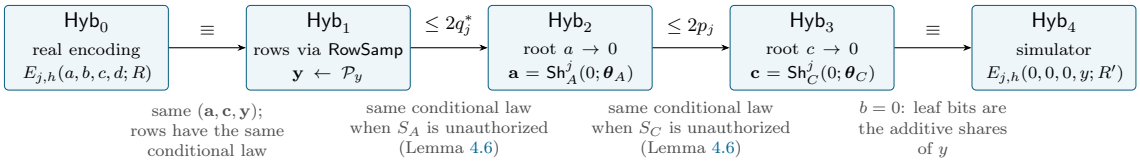


Figure 5: The hybrid argument of Lemma 4.7. The first and last transitions are exact identities; the two middle transitions replace a sharing root by zero and cost at most the probability that the corresponding selected shares are authorized.

Hyb_0. *(The real encoding.)* Sample a uniform random tape R and output the real encoding $\mathsf{Hyb}_0 = E_{j,h}(a,b,c,d;R)$.

Hyb_1. *(Generating the encoding with the row sampler.)* To prepare for changing the sharing roots, we sample the leaf bits directly from their uniform distribution on $\mathcal{P}_y$. Generate $\mathbf{a} = \mathsf{Sh}_A^j(a;\boldsymbol{\theta}_A)$,$\mathbf{c} = \mathsf{Sh}_C^j(c;\boldsymbol{\theta}_C)$ with independent uniform masks, and sample $\mathbf{y} \leftarrow \mathcal{P}_y$ independently of both mask vectors. Given these values, generate h rows at every leaf using $\mathsf{RowSamp}(y_i, c_i, a_i)$ from Equation (6), with fresh independent random bits for every row, and output the resulting matrix list Hyb_1.

By Step 1, the triple $(\mathbf{a},\mathbf{c},\mathbf{y})$ and the conditional distribution of the sampled rows agree with those of the real encoding, so $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_0$.

Hyb_2. *(Replacing the root a by zero.)* We now replace a by zero, keeping the c-sharing unchanged because it determines which a-shares are selected. Generate Hyb_2 as in Hyb_1, except that $\mathbf{a} = \mathsf{Sh}_A^j(0;\boldsymbol{\theta}_A)$, with the leaf bits still sampled uniformly from $\mathcal{P}_y$ and independently of the masks.

Both hybrids generate their matrices by applying the same row-sampling procedure to $(\mathbf{y}, \mathbf{c}[S_C], \mathbf{a}[S_A])$. Since applying the same randomized procedure cannot increase statistical distance, it suffices to compare the distributions of these triples.
The pair $(\mathbf{y}, \mathbf{c}[S_C])$ has the same distribution in both hybrids and, conditioning on this pair, fixes which a-shares are selected, and hence fixes S_A, while leaving the A-sharing masks independent and uniform. The remaining component is therefore $\mathsf{Sh}_A^j(s;\boldsymbol{\theta}_A)[S_A]$, with $s = a$ in Hyb_1 and $s = 0$ in Hyb_2. By Lemma 4.6, these two conditional distributions coincide whenever S_A is unauthorized; when it is authorized, their statistical distance is at most one.

Averaging these conditional distances over the common distribution of $(\mathbf{y}, \mathbf{c}[S_C])$ and applying (9), we obtain

$$\mathrm{SD}(\mathsf{Hyb}_1, \mathsf{Hyb}_2) \leq \mathbb{E}_{\mathbf{y},\mathbf{c}}\big[\mathbf{1}\{\mathrm{Auth}_A^j(S_A) = 1\}\big] = \Pr_{\mathbf{y},\mathbf{c}}[\mathrm{Auth}_A^j(S_A) = 1] \leq 2q_j(c) \leq 2q_j^*,$$

where $\mathbf{y}$ is uniform on $\mathcal{P}_y$ and $\mathbf{c}$ is an independently generated sharing of the original root c.

Hyb_3. *(Replacing the root c by zero.)* Having replaced a by zero, we now do the same for c. Concretely, we generate Hyb_3 as in Hyb_2, except that $\mathbf{c} = \mathsf{Sh}_C^j(0;\boldsymbol{\theta}_C)$, with $\mathbf{y}$ still uniform on $\mathcal{P}_y$ and independent of both mask vectors.

The leaf vector $\mathbf{y}$ has the same distribution in both hybrids, and conditioning on it fixes the set S_C of selected c-shares. By Lemma 4.6, these shares have the same conditional distribution for the roots c and zero whenever S_C is unauthorized; otherwise, their statistical distance is at most one. Changing the selected c-shares may also change which a-shares are selected,

but this requires no additional error bound. Given $(\mathbf{y}, \mathbf{c}[S_C])$, both hybrids use the same randomized procedure: determine S_A, sample an independent A-sharing of zero, restrict it to S_A, and generate the rows with $\mathsf{RowSamp}$ using fresh independent randomness. This procedure includes the selection of the a-shares, so the statistical distance between the matrix lists is at most the distance between the distributions of $(\mathbf{y}, \mathbf{c}[S_C])$.

By convexity of statistical distance, the conditional bounds above also bound the unconditional distance after taking expectation over $\mathbf{y}$; applying (9) then gives

$$\mathrm{SD}(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \le \mathbb{E}_{\mathbf{y}\leftarrow\mathcal{P}_y}\left[\mathbf{1}\{\mathrm{Auth}_C^j(S_C) = 1\}\right] = \Pr_{\mathbf{y}\leftarrow\mathcal{P}_y}\left[\mathrm{Auth}_C^j(S_C) = 1\right] \le 2p_j.$$

Hyb_4. *(The simulator.)* For the final hybrid, we run the simulator on the prescribed output y: sample a fresh uniform random tape R' and output $\mathsf{Hyb}_4 = \mathsf{Sim}_{j,h}(y) = E_{j,h}(0, 0, 0, y; R')$.

On input $(0, 0, 0, y)$, both recursive sharings have root zero, and the leaf bits form a uniform additive sharing of y, independently of the recursive masks. Step 1 therefore gives $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$. Together with the first transition, this give

$$\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1 \qquad \text{and} \qquad \mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4. \tag{10}$$

We now combine these identities with the bounds for the two root replacements. By the triangle inequality,

$$\begin{aligned}\mathrm{SD}\big(E_{j,h}(a, b, c, d; R), \mathsf{Sim}_{j,h}(y)\big) &= \mathrm{SD}(\mathsf{Hyb}_0, \mathsf{Hyb}_4) \\ &\le \mathrm{SD}(\mathsf{Hyb}_1, \mathsf{Hyb}_2) + \mathrm{SD}(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \le 2q_j^* + 2p_j.\end{aligned}$$

Since statistical distance is also at most one, the privacy error is bounded by $\min\{1, 2q_j^* + 2p_j\}$, as claimed. □

Step 4: Completing the privacy proof. We now return to Proposition 4.4. Lemma 4.7 bounds the privacy error by $2p_j + 2q_j^*$, so it remains to bound these probabilities in terms of the sharing depth. Lemma 4.10, proved below using the recursive structure of the two sharings, gives $p_{4+\iota} \le \frac{1}{2}\,5^{-2^\iota}$ and $q_{4+\iota}^* \le \frac{1}{4}\,20^{-2^\iota}$, for every integer $\iota \ge 0$. For $j \ge 4$, we set $\iota = j - 4$ and substitute these estimates into the privacy bound to obtain

$$\begin{aligned}\mathrm{SD}\big(E_{j,h}(a, b, c, d; R), \mathsf{Sim}_{j,h}(abc + d)\big) &\le 2p_j + 2q_j^* \\ &\le 5^{-2^{j-4}} + \tfrac{1}{2}\,20^{-2^{j-4}} \le \tfrac{3}{2}\,5^{-2^{j-4}},\end{aligned}$$

where the last inequality uses $20^{-2^{j-4}} \le 5^{-2^{j-4}}$. This is the claimed privacy bound, and the simulator runs in time $O(Lh)$ because it executes the encoder once. □

Bounding the reconstruction probabilities. We now prove the bounds on p_j and q_j^* used in Step 4 of the proof of Proposition 4.4, by analyzing one sharing level at a time. In a depth-$(j+1)$ sharing with root s, the one-level rule produces four values $x_1, \ldots, x_4$, which we call the *child secrets*. Each x_ν becomes the root of a depth-j subtree, whose 4^j leaf indices form the ν-th *child block*. By Lemma 4.6, the selected shares within each block either determine its child secret or reveal nothing about it. We can therefore analyze reconstruction of s by determining which of the four child secrets can be recovered.

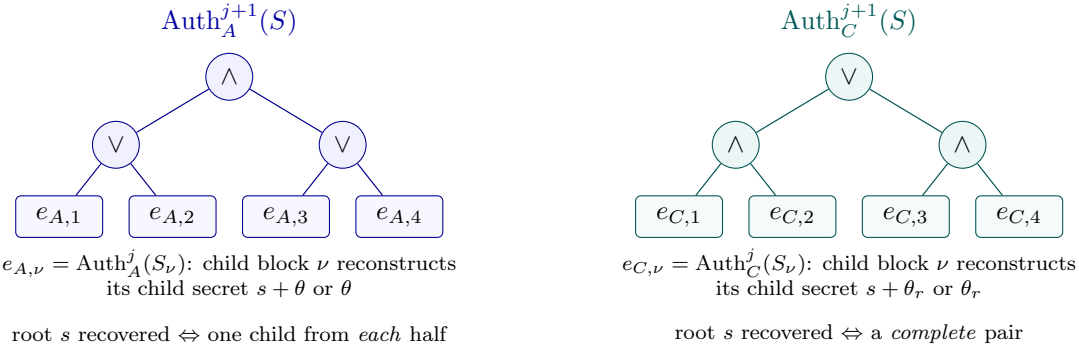


Figure 6: The recursive reconstruction rules of Lemma 4.8. A selected set reconstructs the root of a depth-$(j+1)$ sharing exactly when the child blocks it reconstructs, indicated by the bits e_ν, satisfy the Boolean rule of the one-level scheme: one child from each half for Sh_A, a complete pair for Sh_C.

Lemma 4.8 (Recursive reconstruction rules)**.** *Let $j \geq 0$ and $S \subseteq [4^{j+1}]$. For each $\nu \in [4]$, let $S_\nu \subseteq [4^j]$ contain the indices of the selected leaves in child block ν, numbered within that block. For $X \in \{A, C\}$, let $e_{X,\nu} = \mathrm{Auth}_X^j(S_\nu)$ indicate whether the selected leaves in child subtree ν reconstruct the bit at its root. Thus $e_{X,\nu}$ equals one if reconstruction is possible from those leaves alone, and zero otherwise. Then*

$$\mathrm{Auth}_A^{j+1}(S) = (e_{A,1} \vee e_{A,2}) \wedge (e_{A,3} \vee e_{A,4}),$$
$$\mathrm{Auth}_C^{j+1}(S) = (e_{C,1} \wedge e_{C,2}) \vee (e_{C,3} \wedge e_{C,4}),$$

as illustrated in Figure 6.

Proof. We use the same argument for both sharing schemes. For $X \in \{A, C\}$, let $K = \{\nu \in [4] : e_{X,\nu} = 1\}$ be the set of children whose secrets can be reconstructed from the shares selected by S. For each $\nu \in K$, the selected shares in child block ν reconstruct x_ν, while for $\nu \notin K$ their distribution is independent of x_ν, by Lemma 4.6. Moreover, conditional on the four child secrets, the selected shares in different blocks are independent because the subtrees use fresh independent masks. We can therefore reproduce the selected-share distribution knowing only $(x_\nu)_{\nu \in K}$: independently generate a depth-j sharing in each subtree and keep the shares indexed by S_ν, using x_ν when $\nu \in K$ and zero otherwise. Using zero does not change the distribution in an unauthorized block. In the other direction, the selected shares reconstruct every child secret indexed by K, so it remains to determine which collections of child secrets reconstruct the root.

For the A-sharing, the one-level rule gives $(x_1, x_2, x_3, x_4) = (s + \theta, s + \theta, \theta, \theta)$. Knowing one of x_1, x_2 and one of x_3, x_4 lets us recover s by adding them. If either pair contains no known secret, all the known values are copies of at most one uniform bit, either $s + \theta$ or θ, whose distribution is independent of s. For the C-sharing, we have $(x_1, x_2, x_3, x_4) = s + \theta_1, \theta_1, s + \theta_2, \theta_2)$. Knowing both x_1, x_2 or both x_3, x_4 lets us recover s by addition. If neither pair is complete, we know at most one value from each pair; since θ_1, θ_2 are independent uniform masks, the joint distribution of these known values is independent of s.

These are precisely the two conditions in the statement. When the relevant condition fails, the selected leaf shares can be sampled from child secrets whose joint distribution is independent of s, so they also reveal nothing about s and cannot reconstruct it. □

We now apply Lemma 4.8 to express p_{j+1} and $q_{j+1}(c)$ in terms of the corresponding probabilities at depth j. We continue to use independent uniform bits $\tilde{y}_i$. By Equation (9), conditioning their

sum to equal the prescribed output y increases either reconstruction probability by at most a factor of two.

Lemma 4.9 (Reconstruction probabilities at each depth)**.** *Let p_j and $q_j(c)$ be the probabilities defined in Equation (8), and write $\beta(x) = 1-(1-x)^2 = 2x - x^2$ for $x \in [0,1]$. The initial values are $p_0 = \frac{1}{2}$, $q_0(0) = 0$ and $q_0(1) = \frac{1}{2}$. For every $j \geq 0$,*

$$p_{j+1} = 2p_j^2 - p_j^4, \tag{11}$$

$$q_{j+1}(0) = \left(\frac{\beta(q_j(0)) + \beta(q_j(1))}{2}\right)^2, \tag{12}$$

$$q_{j+1}(1) = \big(q_j(0) + q_j(1) - q_j(0)q_j(1)\big)^2. \tag{13}$$

Proof. We first consider depth zero, where selecting the only leaf reveals the root itself. We select the c-share when $\tilde{y}_1 = 0$, giving $p_0 = 1/2$. For fixed $c = 0$, the a-share is never selected, while for fixed $c = 1$ it is selected exactly when $\tilde{y}_1 = 0$. Hence $q_0(0) = 0$ and $q_0(1) = 1/2$.

To compute p_{j+1}, we consider whether the selected leaf shares reconstruct each of the four child secrets. These events each have probability p_j and are independent because they depend only on the bits $\tilde{y}_i$ in their respective subtrees. Both child secrets in a given pair are therefore reconstructible with probability p_j^2. We recover the root when at least one of the pairs $\{1,2\}$ and $\{3,4\}$ succeeds, giving $p_{j+1} = 1-(1-p_j^2)^2 = 2p_j^2 - p_j^4$.

For the A-sharing, we select a leaf when both $\tilde{y}_i = 0$ and $c_i = 1$, so we must also consider the C-sharing. We condition on its two root masks θ_1, θ_2, fixing its four child secrets to $(c+\theta_1, \theta_1, c+\theta_2, \theta_2)$. The remaining C-sharing masks and the bits $\tilde{y}_i$ are independent across subtrees. Under this conditioning, the four events of reconstructing an A-child secret from its selected leaves are therefore independent, with probability $q_j(t)$ when the corresponding C-child secret is t. To recover the A-root, we need at least one successful child reconstruction in each pair.

When $c = 0$, the first two C-child secrets both equal θ_1, so the corresponding A-reconstruction events each have probability $q_j(\theta_1)$. We recover at least one child unless both reconstructions fail, giving probability $1-(1-q_j(\theta_1))^2 = \beta(q_j(\theta_1))$. The second pair similarly succeeds with probability $\beta(q_j(\theta_2))$. We need both pairs to succeed, so we multiply these conditional probabilities and average over the independent uniform masks to obtain

$$q_{j+1}(0) = \mathbb{E}_{\theta_1,\theta_2}\big[\beta(q_j(\theta_1))\beta(q_j(\theta_2))\big] = \left(\frac{\beta(q_j(0)) + \beta(q_j(1))}{2}\right)^2.$$

When $c = 1$, each pair contains one C-child secret equal to zero and one equal to one. We therefore recover at least one of the corresponding A-child secrets with probability $1-(1-q_j(0))(1-q_j(1)) = q_j(0) + q_j(1) - q_j(0)q_j(1)$. Since the two pairs are independent under the conditioning, we square this probability to require success in both pairs. The expression does not depend on the root masks, so averaging leaves it unchanged and gives (13). □

We now use these formulas to obtain explicit bounds on the reconstruction probabilities as the depth increases. The inequalities $p_{j+1} \leq 2p_j^2$ and $q_{j+1}^* \leq 4(q_j^*)^2$ can be written as

$$2p_{j+1} \leq (2p_j)^2, \qquad 4q_{j+1}^* \leq (4q_j^*)^2.$$

Thus, once $2p_j$ and $4q_j^*$ are below one, each additional sharing level squares their upper bounds, doubling the exponent at every step. To obtain suitable starting bounds, we compute the first four levels using the exact formulas above, which give $2p_4 \leq \frac{1}{5}$ and $4q_4^* \leq \frac{1}{20}$. Applying the squaring inequalities from depth four onwards then gives the bounds used in Proposition 4.4.

Depth j	0	1	2	3	4
p_j	$\frac{1}{2}$	$\frac{7}{16}$	$< \frac{355}{2^{10}}$	$< \frac{463}{2^{11}}$	$< \frac{1}{10}$
q_j^*	$\frac{1}{2}$	$\frac{1}{4}$	$< \frac{65}{2^9}$	$< \frac{29}{2^9}$	$< \frac{1}{80}$

Table 2: Values and upper bounds for the reconstruction probabilities, obtained from Lemma 4.9. The intermediate calculations appear in the proof below. The bounds $p_4 < 1/10$ and $q_4^* < 1/80$ in the last column start the repeated-squaring argument of Lemma 4.10.

Lemma 4.10. *For every integer $s \geq 0$, $p_{4+s} \leq \frac{1}{2}\,5^{-2^s}$ and $q^*_{4+s} \leq \frac{1}{4}\,20^{-2^s}$.*

Proof. We first verify $p_4 < 1/10$ and $q_4^* < 1/80$. Write $F(x) = 2x^2 - x^4$, so that $p_{j+1} = F(p_j)$. Since F is increasing on $[0,1]$, we can iterate rational upper bounds. Starting from $p_0 = 1/2$, we obtain

$$\begin{aligned} p_1 &= \frac{7}{16}, \\ p_2 &= F\Big(\frac{7}{16}\Big) < \frac{355}{2^{10}}, \\ p_3 &< F\Big(\frac{355}{2^{10}}\Big) < \frac{463}{2^{11}}, \\ p_4 &< F\Big(\frac{463}{2^{11}}\Big) < \frac{1}{10}. \end{aligned}$$

Thus we can verify the required bound using these rational upper bounds, without writing out the exact probabilities at every depth.

For q_j^*, we can bound both root values together. The two expressions defining $q_{j+1}(0)$ and $q_{j+1}(1)$ are increasing in each of $q_j(0)$ and $q_j(1)$. Replacing both arguments by q_j^* gives $\beta(q_j^*)^2$ in either expression, and hence $q_{j+1}^* \leq \beta(q_j^*)^2 \leq 4(q_j^*)^2$, where the second inequality uses $\beta(x) = 2x - x^2 \leq 2x$. The initial values give

$$(q_1(0), q_1(1)) = \Big(\frac{9}{64}, \frac{1}{4}\Big), \qquad q_2^* < \frac{65}{2^9}$$

by substitution into the two formulas. Applying the bound with β for the next two levels then gives

$$q_3^* < \beta\Big(\frac{65}{2^9}\Big)^2 < \frac{29}{2^9}, \qquad q_4^* < \beta\Big(\frac{29}{2^9}\Big)^2 < \frac{1}{80}.$$

With these starting estimates established, we no longer need to calculate the probabilities individually. The inequalities $p_{j+1} \leq 2p_j^2$ and $q_{j+1}^* \leq 4(q_j^*)^2$ give

$$2p_{j+1} \leq (2p_j)^2, \qquad 4q_{j+1}^* \leq (4q_j^*)^2.$$

In other words, each additional level squares the upper bound on $2p_j$ and on $4q_j^*$. Since $2p_4 < 1/5$ and $4q_4^* < 1/20$, induction gives $2p_{4+s} \leq (2p_4)^{2^s} \leq 5^{-2^s}$, $4q^*_{4+s} \leq (4q_4^*)^{2^s} \leq 20^{-2^s}$. Dividing by two and four, respectively, proves the lemma. □

Choosing the parameters. For each integer security parameter $\lambda \geq 1$, we instantiate the encoding of $g^\star$ by setting $j_\lambda = 4 + \left\lceil \log_2 \left(\frac{\lambda+2}{\log_2 5} \right) \right\rceil$, $L_\lambda = 4^{j_\lambda}$ and $h_\lambda = \lambda + \lceil \log_2(7L_\lambda) \rceil + 1$, and write $E_\lambda = E_{j_\lambda, h_\lambda}$, $\mathsf{Dec}_\lambda = \mathsf{Dec}_{j_\lambda, h_\lambda}$ and $\mathsf{Sim}_\lambda = \mathsf{Sim}_{j_\lambda, h_\lambda}$. With these parameters, Propositions 4.3 and 4.4 give

$$\delta_\lambda \leq 2^{-\lambda-2}, \qquad \varepsilon_\lambda \leq \frac{3}{2}\,2^{-\lambda-2},$$

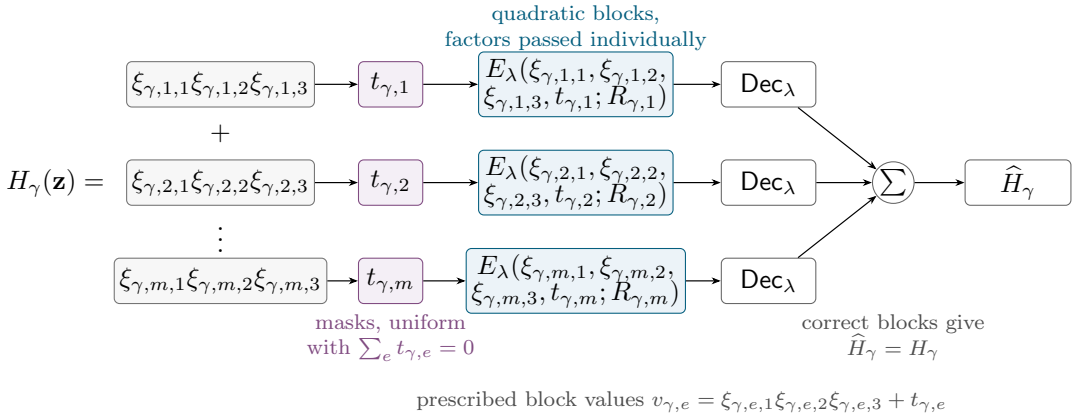


Figure 7: Replacing a cubic coordinate $H_\gamma = \sum_e \xi_{\gamma,e,1}\xi_{\gamma,e,2}\xi_{\gamma,e,3}$ by quadratic encodings (Lemma 4.11). Each monomial contributes one copy of E_λ, whose inputs are the three factors and an additive mask. The masks are uniform subject to summing to zero, so the masked monomials are uniform subject to summing to H_γ. Summing the decoded bits gives an estimate $\widehat{H}_\gamma$, which equals H_γ whenever all copies for this coordinate decode correctly. The full decoder applies Dec_H to the vector $(\widehat{H}_1, \ldots, \widehat{H}_\ell)$ obtained from all coordinates.

where the privacy estimate follows from $(\log_2 5)\, 2^{j_\lambda - 4} \geq \lambda + 2$.

These choices give $L_\lambda = O(\lambda^2)$ and $h_\lambda = O(\lambda)$. By Propositions 4.2 and 4.4, the encoding uses $O(\lambda^3)$ random bits and outputs $O(\lambda^3)$ bits, and its encoder, decoder and simulator run in time $O(\lambda^3)$. Its expanded polynomial description has size polynomial in λ.

4.3 Quadratic encodings of arbitrary functions

We now convert the perfect degree-three encodings of Fact 3.4 into quadratic encodings by masking their monomials and applying our encoding E_λ of $abc + d$ to each masked term, as described in Section 2. The following lemma gives the resulting degree, error and complexity bounds, and Figure 7 shows the construction for one coordinate.

Lemma 4.11 (Replacing cubic coordinates)**.** *Let $H : \mathbb{F}_2^N \times \mathbb{F}_2^r \to \mathbb{F}_2^\ell$ be a δ_0-correct, ε_0-private randomized encoding of f, whose coordinates are explicitly given as sums of monomials of total degree at most three. Let M be the number of these terms, where each zero coordinate is represented by one zero term. For every integer $\lambda \geq 1$, we can construct a randomized encoding of f of degree at most two, with correctness error at most $\delta_0 + M\delta_\lambda$ and statistical privacy error at most $\varepsilon_0 + M\varepsilon_\lambda$. Its expanded polynomial description can be constructed in time polynomial in λ and the description size of H, including $N + r + \ell$. Decoding and simulation require the running time of the corresponding algorithms for H plus $O(M\lambda^3)$.*

If H instead belongs to a computationally private family, the same transformation preserves computational privacy, with the same correctness bound, provided all descriptions and running times are polynomially bounded and $M\varepsilon_\lambda$ is negligible in the security parameter.

Proof. If $\ell = 0$, we keep H. Otherwise, we sample its random tape $\mathbf{w} \leftarrow \mathbb{F}_2^r$ and, independently in each coordinate, choose one mask per term by sampling all but the last uniformly and setting the last to their sum. We write each term as a product of three variables or constants and apply E_λ to these three factors and the term's mask, using fresh independent randomness for every call. All substituted arguments are affine in the original input bits and the independent random bits, so the resulting encoding has degree at most two.

We decode each copy, add the results within each coordinate and apply Dec_H. Conditional on $\mathbf{w}$ and the masks, each copy has a fixed input and fresh randomness, so a union bound gives error at most $M\delta_\lambda$. If no copy errs, we recover $H(\mathbf{x};\mathbf{w})$ exactly, giving total correctness error at most $\delta_0 + M\delta_\lambda$.

Under the same conditioning, replacing the copies by independent calls to Sim_λ on their masked values introduces statistical distance at most $M\varepsilon_\lambda$, also after averaging. Given $H(\mathbf{x};\mathbf{w})$, these masked values can be sampled independently across coordinates by choosing all but the last value uniformly and setting the last to the coordinate plus their sum. Our simulator therefore first runs $\mathsf{Sim}_H(f(\mathbf{x}))$, samples masked values for each returned coordinate in this way, and applies Sim_λ independently to each value. Replacing $H(\mathbf{x};\mathbf{w})$ by its simulation adds at most ε_0, proving the privacy bound. The same efficient procedure proves the computational claim.

The M copies give the stated decoding and simulation overhead. Each affine substitution has at most M terms, so expanding the quadratic descriptions takes polynomial time and gives polynomial description size. □

We now combine Lemma 4.11 with the perfect degree-three encodings of Fact 3.4 to obtain quadratic encodings of arbitrary functions, with complexity polynomial in their branching-program size and the security parameter.

Theorem 4.12 (Quadratic encodings of arbitrary functions)**.** *Let $f : \{0,1\}^N \to \{0,1\}^b$ be represented by a list of mod-2 branching programs of total size σ, including the input and output lengths as in the preliminaries. For every integer $\lambda \geq 1$, f has an explicit randomized encoding of degree at most two over $\mathbb{F}_2$, with correctness error $\delta \leq 2^{-\lambda-2}$ and statistical privacy error $\varepsilon \leq \frac{3}{2}\,2^{-\lambda-2} < 2^{-\lambda}$. Its random tape, encoded output and expanded polynomial description have length polynomial in $\sigma + \lambda$, and its construction, encoder, decoder and simulator run in time polynomial in $\sigma + \lambda$.*

Proof. If $b = 0$, then f always returns the empty string, so we let the encoder, decoder and simulator do the same, using no randomness. We may therefore assume that $b \geq 1$.

We concatenate the encodings of Fact 3.4 for the b output bits, using independent random tapes, to obtain a perfectly correct and perfectly private degree-three encoding H whose expanded polynomial description has size polynomial in σ [AIK06b]. Let $M \geq 1$ be its number of monomials as in Lemma 4.11. Since the error bounds add over the M copies, we choose $\lambda' = \lambda + \lceil \log_2 M \rceil$, so that $M2^{-\lambda'} \leq 2^{-\lambda}$. Applying the lemma with parameter λ' gives a quadratic encoding with correctness error at most $M\delta_{\lambda'} \leq 2^{-\lambda-2}$ and statistical privacy error at most $M\varepsilon_{\lambda'} \leq \frac{3}{2}\,2^{-\lambda-2}$. Since M is polynomial in σ, all lengths and running times are polynomial in $\sigma + \lambda$. □

4.4 Computational encodings for circuits

The preceding theorem gives complexity polynomial in branching-program size, but does not provide a polynomial bound in the size of an arbitrary circuit. For circuit representations, we instead start from the perfectly correct, computationally private degree-three encodings of Applebaum, Ishai and Kushilevitz [AIK06a], whose complexity is polynomial in circuit size under the easy PRG assumption. Applying Lemma 4.11 then gives quadratic encodings with negligible correctness error, while preserving computational privacy and polynomial complexity.

We use the *easy PRG assumption* of Applebaum, Ishai and Kushilevitz [AIK06a]. It requires a secure pseudorandom generator whose output bits can each be computed from the seed by a mod-2 branching program, with both the program size and its construction time polynomial in the seed length. Equivalently, the generator belongs to polynomial-time uniform $\oplus$L/poly. In particular, a polynomial-time uniform PRG in NC^1 suffices.

Application	Property of the 2RE used	Assumption	Where
Local, decomposable 2REs	degree two; one input bit per monomial	none	§5.1
Locality-three one-way functions	statistical errors; degree two	OWF in uniform NC[1]	§5.2
Instance hiding	locality three; privacy	none	§5.3
PSM and CDS	decomposability; locality three	shared tape hidden from referee	§5.3
Single-input garbling	decomposability; locality three	none	§5.3
Local message generation	privacy; decoding	base primitive	§5.4
Circuit-efficient variants	computational 2RE	easy PRG (EPRG)	§5.4

Table 3: Applications of the 2REs and the property each one uses.

Theorem 4.13 (Computational quadratic encodings for circuits)**.** *Under the easy PRG assumption, let (C_λ) be a family of Boolean circuits computing (f_λ), with size $\sigma(\lambda)$ polynomial in λ and including the input and output lengths. There is a computationally private randomized encoding of (f_λ) with correctness error at most $2^{-\lambda-2}$ and total degree at most two jointly in the original input bits and the independent random bits, such that every monomial in its expanded polynomial description contains at most one original input bit. The encoding, decoder and simulator can be constructed from C_λ in time polynomial in $\sigma(\lambda)+\lambda$.*

Proof. Under the easy PRG assumption, we use [AIK06a] to obtain a perfectly correct, computationally private degree-three encoding H_λ of f_λ, with output locality four and complexity polynomial in $\sigma(\lambda)+\lambda$. To handle arbitrary circuit families, we apply the cited construction to a fixed polynomial-time uniform family of circuit evaluators and then fix the public circuit description and input padding. We choose the padded length m large enough for the description, input and output, with $\lambda \le m \le \mathrm{poly}(\sigma(\lambda)+\lambda)$. The decoder removes the output padding, and the simulator adds it before simulation. The polynomial relation between m and λ preserves computational privacy against nonuniform distinguishers.

Locality four lets us expand H_λ into polynomially many monomial terms. Let M be the number of these terms as in Lemma 4.11, and set $\lambda' = \lambda + \lceil \log_2 \max\{1, M\} \rceil$. Applying that lemma gives a quadratic encoding with correctness error at most $M\delta_{\lambda'} \le 2^{-\lambda-2}$. Computational privacy is preserved because the additional statistical error is at most $M\varepsilon_{\lambda'} \le \frac{3}{2}\,2^{-\lambda-2}$, which is negligible.

By Equation (2), each monomial of $E_{\lambda'}$ contains at most one of a, b, c, d. We substitute individual variables or constants for a, b, c and a linear mask depending only on randomness for d, so every resulting monomial contains at most one original input bit. The complexity bounds follow from Lemma 4.11. □

5 Applications of Quadratic Randomized Encodings

We now use the quadratic encodings of Section 4 to obtain locally computable one-way functions, protocols for private computation and simpler cryptographic message-generation algorithms. These applications use established reductions [AIK06b, AIK06a, Ish13], and for each one we state the assumptions and explain how the encoding errors affect correctness and security. Table 3 summarizes the applications.

5.1 Locality three and decomposability

Applying the masking transformation of Applebaum, Ishai and Kushilevitz [AIK06b] to our quadratic encodings gives output locality three, resolving the question posed by Ishai [Ish13] with arbitrarily

small correctness and statistical privacy errors. The same transformation also gives decomposability for our encodings, as we show below.

An encoding is *decomposable* [Ish13] if its output coordinates can be grouped as

$$E_f(\mathbf{x}; R) = \big(E_0(R), E_1(x_1; R), \ldots, E_N(x_N; R)\big),$$

where each E_i uses only x_i and the random tape R, while E_0 uses only R. Blocks may be empty and may share random bits. Thus separate input holders can compute their respective blocks using their own inputs and a shared tape.

Corollary 5.1 (Decomposable statistical REs of locality three)**.** *Let $f : \{0,1\}^N \to \{0,1\}^b$ be represented by a list of mod-2 branching programs of total size σ, including the input and output lengths. For every $\lambda \geq 1$, f has a decomposable statistical RE of degree at most two and output locality at most three, with correctness error at most $2^{-\lambda-2}$, privacy error less than $2^{-\lambda}$, and complexity polynomial in $\sigma + \lambda$.*

Proof. We can apply the locality transformation of [AIK06b, Construction 4.16 and Lemma 4.17] to the quadratic encoding of Theorem 4.12. This transformation is perfectly correct and perfectly private, so composition preserves both encoding errors, while the explicit polynomial description gives polynomial overhead.

To check decomposability, recall that the starting degree-three encoding has at most one original input bit in each monomial, by Fact 3.4. Lemma 4.11 replaces each such monomial by a copy of the masked-product encoder, whose arguments are the individual factors and a mask depending only on randomness, so the entire copy depends on at most one original input bit. The locality transformation masks each resulting monomial separately, and its remaining output bits depend only on randomness. At this point, grouping these coordinates by their original input bit gives the required decomposable form. □

The same argument applies to the computational encodings of Section 4.4, giving complexity polynomial in circuit size under the easy PRG assumption used there.

Corollary 5.2 (Computational local 2REs)**.** *Under the PRG assumption of Section 4.4, every polynomial-size circuit family has a computationally private, decomposable 2RE of output locality at most three, with correctness error at most $2^{-\lambda-2}$ and complexity polynomial in the circuit size and the security parameter λ.*

Proof. The encodings of Theorem 4.13 also have at most one original input bit in each monomial, so the preceding locality and decomposability argument applies with polynomial overhead. Since the locality transformation is perfectly correct, its decoder recovers the original encoding without error, and hence the correctness error is unchanged.

By perfect privacy of the locality transformation, the transformed encoding has exactly the distribution obtained by applying its simulator to the original encoded string. We can therefore simulate the final encoding by first running the original simulator on the function value and then applying the locality simulator with fresh randomness. Computational privacy is preserved because any efficient algorithm distinguishing the resulting distributions could be composed with the locality simulator to distinguish an original encoding from its simulation. □

5.2 One-way functions with optimal output locality

Applebaum, Ishai and Kushilevitz obtained one-way functions of output locality four from one-way functions in uniform NC^1 [AIK06b]. Combining our statistical 2REs with their reductions [AIK06b],

we obtain output locality three under the same assumption. This locality is optimal, since a map of output locality two can be inverted on its image by solving a 2-SAT instance [AIK06b].

Decoding errors and one-wayness. To obtain a one-way function, we might try to use the encoding map itself, treating both its input and its random tape as the input to a deterministic function. With perfect correctness, all pairs producing the same encoded value must have the same original function value, since the decoder recovers that value from the encoding alone. Unfortunately, with statistical correctness, an inverter may return a pair on which decoding fails, even if such pairs are rare when the input and tape are sampled honestly.

We therefore use the approach of [AIK06b], which first establishes *distributional one-wayness*.

From statistical encodings to distributional one-wayness. The following lemma applies to statistical encodings, allowing negligible errors in both correctness and privacy.

Lemma 5.3 (Statistical encodings of one-way functions [AIK06b])**.** *Let (f_λ) be a one-way function family, and let $E_\lambda(\mathbf{x};\mathbf{r})$ be an efficient randomized encoding of f_λ whose tape consists of $\rho(\lambda)$ independent uniform bits. Suppose its correctness error δ_λ and statistical privacy error ε_λ are negligible. Then $g_\lambda(\mathbf{x},\mathbf{r}) = E_\lambda(\mathbf{x};\mathbf{r})$, viewed as a deterministic function of the full input $(\mathbf{x},\mathbf{r})$, is distributionally one-way.*

Converting to an ordinary one-way function. To obtain ordinary one-wayness while keeping degree two, we use the transformation of Applebaum, Ishai and Kushilevitz [AIK06b], based on the conversion of Impagliazzo and Luby [IL89]. Together with their locality transformation, it gives the following fact.

Fact 5.4 ([AIK06b])**.** *If there exists a polynomial-time uniform family of distributionally one-way functions of degree at most two over $\mathbb{F}_2$, then there exists a polynomial-time uniform family of one-way functions of degree at most two and output locality at most three. The transformation has polynomial overhead.*

When the encoding in Lemma 5.3 is quadratic jointly in its input and random bits, these bits together form the input to a quadratic distributionally one-way function. We can therefore apply Fact 5.4 to our statistical encodings and obtain the following theorem.

Theorem 5.5 (One-way functions with output locality three)**.** *If one-way functions exist in polynomial-time uniform NC^1, then there exists a polynomial-time uniform family of one-way functions of degree at most two over $\mathbb{F}_2$ and output locality at most three.*

Proof. We start with a one-way function family f_λ computed by polynomial-time uniform NC^1 circuits. To apply Theorem 4.12 efficiently, we need mod-2 branching programs of polynomial total size for its output bits. Since the circuits have bounded fan-in and logarithmic depth, duplicating reused subcircuits gives a polynomial-size formula for each output bit. The standard conversion from these formulas to mod-2 branching programs is uniform and has polynomial overhead [AIK06b], giving the required representation.

Applying Theorem 4.12 to this representation, we obtain a quadratic randomized encoding E_λ of f_λ with correctness and statistical privacy errors at most $2^{-\lambda}$. We regard both its input and its random tape as the input to the deterministic function $G_\lambda(\mathbf{x},\mathbf{r}) = E_\lambda(\mathbf{x};\mathbf{r})$. The degree bound for E_λ holds jointly in these variables, so G_λ has degree at most two. Moreover, the encoding is efficient and both errors are negligible, so Lemma 5.3 applies and shows that G_λ is distributionally one-way. We now apply Fact 5.4 to obtain a one-way function family of degree at most two and output locality at most three. The branching-program conversion, our encoding construction and

the conversion in that fact are all uniform and have polynomial overhead, so the resulting family is polynomial-time uniform, as claimed. □

5.3 Private delegation and protocols with shared randomness

We now apply our decomposable statistical encodings to private delegation, private simultaneous-message protocols, conditional disclosure of secrets and garbling for a single input, following standard reductions [AIK06a, Ish13, GIKM00]. In each application, we use the encoding's decoder to recover the required value and its simulator to prove privacy.

Throughout this subsection, let $f : \{0,1\}^N \to \{0,1\}^b$ have a public mod-2 branching-program representation of total size σ, including the input and output lengths, and let $\lambda \geq 1$ be an integer. We use the encodings of Corollary 5.1, with $\delta = 2^{-\lambda-2}$ and $\varepsilon = 2^{-\lambda}$ as upper bounds on their correctness and statistical privacy errors. The maps producing the client queries, sender messages and garbling data described below have degree at most two and output locality at most three, measured jointly in their input bits and independent random bits.

Single-server instance hiding. A client holding x wishes to delegate the computation of $f(x)$ to a server while keeping both its input and the result private. Following [AIK06a], we use the function $h(x, z) = f(x) + z$, where $z \in \mathbb{F}_2^b$ masks the function value. This function has a branching-program representation of size polynomial in σ, so Corollary 5.1 gives an encoder E_h and a decoder Dec_h for h. The client samples a uniform mask $z \in \mathbb{F}_2^b$ and an independent uniform encoding tape R, then sends

$$Q = E_h(x, z; R)$$

to the server while keeping z private. The server applies the decoder and returns $A = \mathsf{Dec}_h(Q)$. If A is a b-bit string, the client outputs $A + z$; if the decoder reports failure, the client also reports failure. When the server follows the protocol and decoding is correct, $A = f(x) + z$, so the client recovers $f(x)$. The client does not verify the correctness of the server's answer.

Corollary 5.6 (Single-server instance hiding)**.** *The resulting single-query delegation protocol has complexity polynomial in $\sigma + \lambda$. When the server follows the protocol, the client recovers $f(x)$ with correctness error at most δ. For every input x, the query distribution is within statistical distance less than ε of a fixed distribution independent of x.*

Private simultaneous messages. In the private simultaneous-message model of Feige, Kilian and Naor [FKN94], the bits of an input $x = (x_1, \ldots, x_N)$ are partitioned among $k \geq 1$ senders. The senders share a uniform random tape R, sampled independently of their inputs for each execution and hidden from the receiver. Each sender sends one message to the receiver without communicating with the other senders, and the receiver should recover $f(x)$ without learning further information about x.

Following [Ish13], we construct the messages using the decomposable encoding $E_f(x; R) = \big(E_0(R), E_1(x_1; R), \ldots, E_N(x_N; R)\big)$. Each sender includes the blocks corresponding to its input bits in its message, which it can compute using only those bits and the shared tape. One designated sender also includes $E_0(R)$. The receiver arranges the blocks in their original order, obtaining exactly $E_f(x; R)$, and applies its decoder. To simulate the messages jointly, we run the encoding simulator on $f(x)$ and group its output into sender messages in the same way, preserving the privacy error.

Corollary 5.7 (Private simultaneous messages)**.** *For any partition of the input bits among $k \geq 1$ senders, the resulting protocol has complexity polynomial in k, σ, λ. The receiver's correctness error is at most δ, and the joint message distribution is simulatable from $f(x)$ with statistical error less than ε.*

Here privacy is required for the receiver, whose view consists of the messages and does not include the shared tape R. Privacy against coalitions of parties that also know their own random tapes is treated in Section 6.1, where we use multiparty randomized encodings with independent local tapes.

Garbling for a single input. Garbling allows one party, the garbler, to prepare the data needed to evaluate a public function f before knowing the input. The garbler produces a public string and two labels for each input bit, one corresponding to zero and the other to one. The standard construction from decomposable encodings $E_f(x;R) = (E_0(R), E_1(x_1;R), \ldots, E_N(x_N;R))$ sets

$$G = E_0(R), \qquad K_i^0 = E_i(0;R), \qquad K_i^1 = E_i(1;R), \qquad i \in [N],$$

using a uniform tape R. For an input $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, where x_i is the ith input bit, the evaluator receives G together with the corresponding label $K_i^{x_i}$ for each i. Thus the resulting tuple $(G, K_1^{x_1}, \ldots, K_N^{x_N})$ is exactly $E_f(x;R)$. Each label is obtained by fixing an input bit of an encoding block to zero or one, which cannot increase either its degree or its locality. Thus the complete map from R to G and all the labels has degree at most two and output locality at most three.

Corollary 5.8 (Single-input private garbling)**.** *For every input x fixed independently of the garbling randomness, an evaluator given only the selected tuple recovers $f(x)$ with correctness error at most δ, and this tuple can be efficiently simulated from $f(x)$ with statistical error less than ε. All algorithms run in time polynomial in $\sigma + \lambda$, and the total output length is polynomial in $\sigma + \lambda$.*

Conditional disclosure of secrets. In conditional disclosure of secrets [GIKM00], two senders hold inputs a and b, respectively, and the first sender also holds a secret bit s. A receiver already knows both a and b and receives one message from each sender. These messages should allow it to recover s when $P(a,b) = 1$ for a public predicate P, while revealing essentially nothing about s when $P(a,b) = 0$.

Let $P : \{0,1\}^{n_A} \times \{0,1\}^{n_B} \to \{0,1\}$ have a mod-2 branching-program representation of size σ_P. We apply the private simultaneous-message construction to

$$h(s,a,b) = sP(a,b),$$

with (s,a) held by the first sender and b by the second. A branching program for h is obtained by adding a new initial vertex and an edge labelled s leading to the original initial vertex. This multiplies the value computed by the program by s and adds only one vertex and one edge, so Corollary 5.7 applies. The senders use a fresh uniform shared tape, independent of (s,a,b) and hidden from the receiver, who decodes their messages to recover $h(s,a,b)$.

When $P(a,b) = 1$, the decoded value is s whenever decoding succeeds. When $P(a,b) = 0$, the function value is zero for either secret, so we simulate the joint messages by running the PSM simulator on zero. Including the receiver's fixed inputs (a,b) in both the real and simulated views preserves the statistical distance.

Corollary 5.9 (Conditional disclosure of a secret)**.** *The resulting protocol has complexity polynomial in $n_A + n_B + \sigma_P + \lambda$. For every fixed (s,a,b), when $P(a,b) = 1$, the receiver recovers s with correctness error at most δ; when $P(a,b) = 0$, the joint distribution of its messages can be simulated from (a,b) alone with statistical error less than ε.*

5.4 Local generation of cryptographic messages

We now use our encodings to make cryptographic message generation local: each transmitted bit depends on at most three input and random bits, even when the original message-generation

algorithm is more complex. Following the transformations of [AIK06b, AIK06a], the sender evaluates the encoding directly, without first computing the original message, and the receiver decodes before applying the original decryption or verification algorithm.

Let F be the original message-generation algorithm, and let z contain all its inputs, including its random coins, so that $F(z)$ is determined once z is fixed. We replace this algorithm by

$$\widehat{F}(z; R) = E_F(z; R),$$

which the sender evaluates directly on z, using a fresh independent encoding tape R for each message. The transmitted string is therefore an encoding of the original message $F(z)$.

The receiver first applies Dec_F to recover the original message and then runs the original decryption or verification algorithm, rejecting malformed strings and the decoder's failure symbol $\perp$. For commitments, this means checking the original opening information against the decoded commitment [AIK06b]. The additional tape R is used only for this encoding and remains private, including during later openings or responses. This does not prevent the sender from revealing the original algorithm's coins when required by the underlying protocol, since those coins are part of z and are independent of R.

Corollary 5.10 (Local generation of cryptographic messages)**.** *Suppose the honest message-generation algorithm of a cryptographic primitive, with its random coins included as inputs, is represented by polynomial-size mod-2 branching programs. Applying the transformation above with the encoding of Corollary 5.1 gives a message-generation map of degree at most two and output locality at most three, with polynomial overhead in the representation size and the security parameter. The transformed primitive preserves the following security properties:*

(i) Indistinguishability under chosen-plaintext attack for public-key or stateless secret-key encryption;

(ii) Existential unforgeability under chosen-message attack for stateless signatures and message-authentication codes, where a forgery must concern a message not previously queried;

(iii) Statistical or computational hiding, and perfect, statistical or computational double-opening binding, for non-interactive commitments;

(iv) Statistical or computational zero knowledge, and the original soundness guarantee, for non-interactive zero-knowledge proofs or arguments with an efficient honest prover.

Each honestly generated message adds at most $2^{-\lambda-2}$ to the original correctness or completeness error, while double-opening binding and soundness are preserved without loss. The degree and locality bounds hold jointly in all inputs to message generation, including keys, messages, public inputs, witnesses and original random coins, together with the fresh encoding bits. Key generation and setup remain unchanged.

For polynomial-size circuit implementations, we obtain the corresponding transformation from Corollary 5.2 under the easy PRG assumption of Section 4.4, with overhead polynomial in circuit size and the security parameter. In this case, hiding and zero knowledge are computational, while double-opening binding and soundness retain their original strength.

Proof. The degree, locality and complexity bounds follow from Corollary 5.1, with the original coins included among the inputs to F. Let us use $\delta \leq 2^{-\lambda-2}$ and $\varepsilon < 2^{-\lambda}$ for the encoding's errors. Whenever decoding recovers $F(z)$, the receiver performs the original decryption or verification procedure, so each honest message adds at most δ to the original correctness or completeness error.

For security, we replace each honest message $E_F(z; R)$ by $\mathsf{Sim}_F(F(z))$. Since every encoding tape is fresh and independent, Lemma 3.3 bounds the change in any fixed security experiment involving at most q honest messages by $q\varepsilon$, even when messages are generated adaptively. The comparison includes the original inputs and coins, so later disclosures required by the original protocol are also covered, while the additional encoding tapes remain private and are not reused.

After these replacements, the transmitted strings can be generated from the original messages alone. Encryption security and commitment hiding therefore follow from the original guarantees under efficient randomized processing. For zero knowledge, we apply Sim_F to the proof produced by the original zero-knowledge simulator. The additional error is negligible for polynomially many messages.

For unforgeability, we answer signing or authentication queries by applying Sim_F to the original oracle's replies. Any accepted encoded forgery decodes to an accepted original signature or tag on the same new message, giving a forgery against the original scheme with only the loss from simulating the replies. Binding and soundness follow directly from verification after decoding: two accepted openings to different values give two such openings of the same decoded commitment, and an accepted proof of a false statement decodes to an accepted original proof of that statement. These implications hold for arbitrary received strings, so double-opening binding and soundness are preserved without loss.

For circuit implementations, we use Corollary 5.2. The same replacements preserve computational indistinguishability, giving computational hiding and zero knowledge. The correctness bound and the decoding arguments remain unchanged. □

6 Quadratic Multiparty Encodings and Their Applications

We construct quadratic multiparty encodings with statistical privacy against every coalition, together with computationally private variants whose complexity is polynomial in the circuit size. These encodings allow us to reduce secure computation to calls to a three-party functionality, which we implement using oblivious transfer; for functionalities with public outputs, the resulting protocols also provide weakly-active privacy. We then apply the encodings in the client–server setting, where the servers evaluate the public map on shares of the clients' local values, allowing the receiver to reconstruct and decode the encoded output.

6.1 Fully private multiparty randomized encodings

To extend the ordinary encodings of Section 4 to the multiparty setting, we need privacy even when the adversaty knows the complete local random tapes of all corrupted parties. We use the conversion of [ABT18], which also allows statistical errors [AIKP22], and let every party contribute to the randomness used for each recipient's encoding.

Let $f = (f_1, \ldots, f_k)$, $x = (x_1, \ldots, x_k)$ be the parties' inputs, and let E_j be an ordinary encoder for f_j. Each party P_i samples a uniform string $s_{i,j}$ for every recipient P_j, of the same length as the random tape of E_j. We choose all these strings independently, and those sampled by P_i form its local random tape. For each recipient P_j, the public map adds the parties' shares and uses their sum as the random tape of E_j, computing

$$r^{(j)} = \sum_{i=1}^{k} s_{i,j}, \text{ so that } \hat{y}_j = E_j(x; r^{(j)}),$$

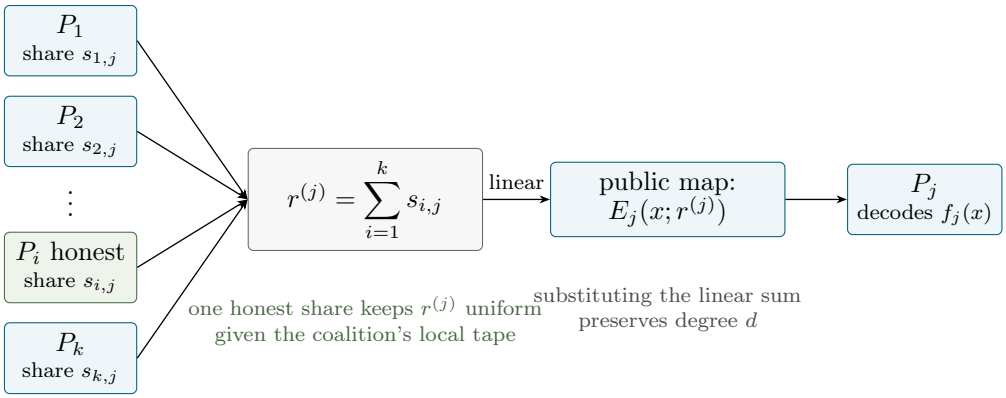


Figure 8: Encoding the output of P_j by sharing the random tape among all parties. Any party outside the coalition contributes an independent uniform share, so the combined tape remains uniform conditional on the coalition's local tapes.

with addition coordinatewise over $\mathbb{F}_2$. It delivers $\hat{y}_j$ only to P_j, who applies the decoder for E_j. Since the parties supply all the random bits, the public map needs no internal randomness. Figure 8 shows the construction for one recipient.

Proposition 6.1 (Degree-preserving multiparty conversion)**.** *Let $k \geq 1$ and let $f = (f_1, \ldots, f_k)$ be a deterministic k-party functionality with bit-string inputs and outputs. Suppose that each nonempty output $f_j(x)$ has an RE $E_j(x; r^{(j)})$ over $\mathbb{F}_2$ of degree at most d, with correctness error δ_j and privacy error ε_j. For a party with no output, use the empty encoding, with randomness length and errors zero. Then the conversion above gives an MPRE without internal randomness whose public map has total degree at most d jointly in the original input bits and the independent local random bits, and hence effective degree at most d. Each party j has correctness error at most δ_j, and every coalition $T \subseteq [k]$ has privacy error at most $\sum_{j \in T} \varepsilon_j$. The conversion has polynomial overhead.*

Proof. Each combined tape $r^{(j)}$ is uniform, so P_j receives an ordinary encoding of $f_j(x)$ with correctness error at most δ_j. Since the bits of $r^{(j)}$ are linear sums of local random bits, substituting them into E_j preserves degree at most d.

For privacy, we first consider a proper coalition T and fix its complete local tapes, including its shares for recipients outside T. Each combined tape still contains a uniform share from an honest party, and these honest shares are independent across recipients. Thus the combined tapes remain independent and uniform even after we fix the corrupted parties' randomness.

Given the coalition's inputs and prescribed outputs, we simulate its view by sampling its complete local tapes and independently running the ordinary simulator for each E_j, with $j \in T$, on $f_j(x)$. For every fixed choice of the coalition's tapes, the received encodings have their ordinary distributions, so replacing them one at a time by these simulations changes the joint distribution by at most $\sum_{j \in T} \varepsilon_j$. Averaging over the tapes gives the same bound for the coalition's complete view.

If all parties are corrupted, the simulator knows the full input and reproduces the execution exactly by sampling all local tapes and evaluating the public map. All randomness is supplied locally, and the construction uses k shares per original tape and one evaluation of each encoder, giving polynomial overhead. □

Corollary 6.2 (Fully private quadratic MPREs)**.** *Let $n \geq 1$ and let f be a deterministic n-party functionality with bit-string inputs and outputs, represented by mod-2 branching programs of total size σ, including the input and output lengths. For every integer $\lambda \geq 1$, f has a fully private 2MPRE*

over $\mathbb{F}_2$, without internal randomness, with statistical privacy error at most $\frac{3}{2}\,2^{-\lambda-2} < 2^{-\lambda}$. On every input, the probability that any party decodes incorrectly is at most $2^{-\lambda-2}$. Its construction, local maps, public map, decoders and simulator run in time polynomial in $n+\sigma+\lambda$, as is its total encoding length. If only one party has a prescribed output, only that party receives a nonempty encoding.

Proof. We apply Theorem 4.12 to each nonempty output vector f_j, using parameter $\lambda' = \lambda + \lceil \log_2 n \rceil$, and then apply Proposition 6.1. Each ordinary encoding has correctness error at most $2^{-\lambda'-2}$ and privacy error at most $\frac{3}{2}\,2^{-\lambda'-2}$. Since $n2^{-\lambda'} \le 2^{-\lambda}$, summing the privacy errors over any coalition gives at most $\frac{3}{2}\,2^{-\lambda-2}$. A union bound over all parties similarly gives probability at most $n2^{-\lambda'-2} \le 2^{-\lambda-2}$ that any party decodes incorrectly.

The conversion preserves degree two and uses no internal randomness, while parties without outputs receive empty encodings. Since we increase the security parameter by only $\lceil \log_2 n \rceil$, all lengths and running times remain polynomial in $n+\sigma+\lambda$. □

6.2 Circuit-efficient computational multiparty encodings

Corollary 6.2 gives statistical multiparty encodings with complexity polynomial in branching-program size. We now obtain complexity polynomial in circuit size, allowing computational privacy and assuming only the existence of a secure PRG.

We use the non-interactive reduction of [ABG+20], in the form described in [AIKP22], which allows the parties to compute their prescribed outputs using local computations and parallel calls to a fixed four-party functionality. We then replace each call by a fully private quadratic MPRE of that functionality. To carry out this construction, we first define the four-party functionality and show how our encoding of $abc+d$ gives the required MPRE.

Definition 6.3 (The functionality $f^\star$)**.** Let $f^\star$ be the deterministic four-party functionality over $\mathbb{F}_2$ in which A, B, C hold $(a,u), (b,v), (c,w) \in \mathbb{F}_2^2$, respectively, and receive no output, while D has no input and receives

$$f^\star{}_D(\mathbf{x}) = abc + u + v + w, \qquad \mathbf{x} = ((a,u),(b,v),(c,w),\perp).$$

We also write $f^\star(\mathbf{x})$ for this receiver's output. This is the designated-receiver functionality $\mathrm{CubePlus}_4^R$ of [AIKP22].

Substituting $d = u+v+w$ in the encoder E_λ of Section 4 preserves degree two because the substitution is linear. Sharing its tape by Proposition 6.1 therefore gives a 2MPRE of $f^\star$ with correctness error $\delta_\lambda \le 2^{-\lambda-2}$, statistical privacy error $\varepsilon_\lambda \le \frac{3}{2}\,2^{-\lambda-2}$, no internal randomness, and an encoded output delivered only to D. We call this the $f^\star$-encoding at parameter λ.

To use our $f^\star$-encoding for a general functionality f, we first express the computation of f using local computations and parallel calls to $f^\star$. Each call has four roles, A, B, C, D, which we assign to the parties computing f; a party may play several roles. We define this form of reduction for an arbitrary target functionality g, whose number of roles we denote by n'.

Definition 6.4 (Non-interactive reductions [ABT18, AIKP22])**.** Let f be a deterministic n-party functionality and g a deterministic n'-party functionality. A non-interactive reduction Π from f to g lets the n parties compute their prescribed outputs using M parallel calls to g and local computations, as follows.

(i) *Assigning the roles.* For each call $j \in [M]$, a fixed public map $\pi_j : [n'] \to [n]$ assigns each role a of g to a party $P_{\pi_j(a)}$. That party supplies the input and receives the output for the assigned role. One party may play several roles, so π_j need not be injective.

(ii) *Preparing the inputs.* Each party P_i independently samples a uniform local tape $r_i \in R_i$, where R_i is finite and nonempty. Using only its own input x_i and tape r_i, it computes the inputs for all its assigned roles in all calls. We write $\mathbf{u}^{(j)}$ for the resulting input tuple of call j.

(iii) *Computing the outputs.* Each call j computes $\mathbf{y}^{(j)} = g(\mathbf{u}^{(j)})$ and delivers $y_a^{(j)}$ only to the party assigned role a. Each party then computes its final output from its original input, its local tape and the call outputs it receives.

The number of calls and their role assignments are fixed as part of the reduction. All call inputs are prepared before any call outputs are received, and there is no other communication. Let $\mathbf{r} = (r_1, \dots, r_n)$, the view of a coalition $T \subseteq [n]$ is

$$\mathrm{View}_T^{\Pi}(\mathbf{x}) = \big(\mathbf{x}[T], \mathbf{r}[T], (y_a^{(j)})_{j \in [M],\, a \in [n']:\, \pi_j(a) \in T}\big),$$

which includes the corrupted parties' inputs, complete local tapes and all call outputs delivered to their assigned roles.
The reduction is δ-correct if, for every input $\mathbf{x}$ and every party P_i, its final output equals $f_i(\mathbf{x})$ with probability at least $1 - \delta$. It is (τ, ε)-private if there is a randomized simulator Sim such that

$$\mathsf{Sim}(T, \mathbf{x}[T], f_T(\mathbf{x})) \approx_\varepsilon \mathrm{View}_T^{\Pi}(\mathbf{x})$$

for every input $\mathbf{x}$ and every coalition $T \subseteq [n]$ with $|T| \le \tau$. It is perfect if $\delta = \varepsilon = 0$, and fully private if $\tau = n$. Efficiency and computational privacy follow the conventions of Definition 3.7.

The completeness theorem of [ABG+20], as used in [AIKP22], gives non-interactive reductions in which the parties make parallel calls to a functionality that computes $abc + u + v + w$. In each call, specified parties supply the input pairs $(a, u), (b, v), (c, w)$, and the result is delivered to all participating parties. For circuit representations, the reduction uses any secure PRG as a black box and has perfect correctness and computational privacy against every coalition, with complexity polynomial in the circuit size, the number of parties and the security parameter. For branching-program representations, it has perfect correctness and privacy, with complexity polynomial in the representation size and the number of parties.

We obtain a reduction to $f^\star$ by replacing each such call with parallel calls to $f^\star$, one for each party that should receive the result. The parties supply the same three input pairs in every replacement call, while we choose a different recipient as the designated receiver D in each call. A party may supply more than one input pair and may also be the receiver. Since the parties use their original local tapes and each recipient obtains the same bit as before, the replacement preserves correctness and privacy.

Remark 6.5 (Designated receivers)**.** If P_d is the only party with a prescribed output, we can omit every call to $f^\star$ whose receiver is another party. Since all call inputs are prepared before any outputs are received, and P_d receives none of the omitted outputs, this leaves the remaining calls and P_d's computation unchanged. All other parties return their prescribed empty outputs, and we preserve privacy by deleting the omitted call outputs from the simulated views.

We now replace each call to $f^\star$ by a quadratic MPRE of $f^\star$. We use the MPRE composition argument of [ABT18, Lemmas 1 and 2], keeping track of both correctness and statistical privacy errors.

Lemma 6.6 (Composition)**.** *Let Π be a non-interactive reduction from a deterministic n-party functionality f to $f^\star$ using M calls, with correctness error δ_0 and statistical privacy error at most*

ε_0 against a coalition $T \subseteq [n]$. Let E be a 2MPRE of $f^\star$ over $\mathbb{F}_2$, with correctness error δ_1 and statistical privacy error at most ε_1 against every coalition. Replacing each call by a copy of E, using fresh randomness independent across copies and of the randomness of Π, gives a 2MPRE of f with correctness error at most $\delta_0 + M\delta_1$ per party and statistical privacy error at most $\varepsilon_0 + M\varepsilon_1$ against T. The resulting encoding has no internal randomness if E has none, and all algorithms have polynomial overhead in the descriptions and running times of Π and the M copies of E. For computationally private reduction families, the same construction preserves computational privacy when all algorithms and simulators run in polynomial time and $M\varepsilon_1$ is negligible, under the conventions of Definition 3.7.

The proof appears in Appendix B. We now combine the computationally private reduction to $f^\star$ with our statistical $f^\star$-encoding to obtain the following result.

Corollary 6.7 (Circuit-efficient computational 2MPREs)**.** *Assume that secure pseudorandom generators exist. There is a construction that, given a Boolean circuit of size σ for a deterministic n-party functionality f and an integer $\lambda \geq 1$, produces a 2MPRE of f over $\mathbb{F}_2$, without internal randomness, with correctness error at most $2^{-\lambda-2}$ for each party. Here σ includes the input and output lengths.*

The construction has computational privacy against every coalition for polynomially bounded families in the sense of Definition 3.7. It uses the generator as a black box, and its total encoding length and all running times, including construction time, are polynomial in $n + \sigma + \lambda$. If only one party has a prescribed output, only that party receives a nonempty encoded output.

Proof. We use the perfectly correct, computationally private reduction of [ABG+20] at security parameter λ and convert its calls to $f^\star$ as described above. If f has only one output party, we first remove the calls whose outputs go to other parties, as justified by Remark 6.5. Writing M for the number of remaining calls, we replace each call by an independent copy of our statistical $f^\star$-encoding at parameter $\lambda' = \lambda + \lceil \log_2 \max\{1, M\} \rceil$, chosen so that $M2^{-\lambda'} \leq 2^{-\lambda}$.

Since the reduction is perfectly correct, Lemma 6.6 gives correctness error at most $M2^{-\lambda'-2} \leq 2^{-\lambda-2}$ for each party. The same lemma bounds the additional statistical privacy error by $\frac{3}{2}M2^{-\lambda'-2} \leq \frac{3}{2}\,2^{-\lambda-2}$, which is negligible in λ, so we preserve the computational privacy of the reduction.

We obtain effective degree at most two and no internal randomness because our $f^\star$-encoding has both properties and the composition preserves them. The reduction uses polynomially many calls, while each encoding has complexity polynomial in λ', so all lengths and running times remain polynomial in $n + \sigma + \lambda$. We also preserve the black-box use of the PRG, since our statistical encodings require no computational assumptions. Finally, when f has only one output party, every retained call has that party as its receiver, and our $f^\star$-encoding delivers a nonempty encoded output only to that receiver. □

Here we evaluate the PRG during the parties' local preprocessing, whose degree is unrestricted, and then apply the quadratic public map to the resulting local values. We can therefore use any secure PRG. For the ordinary encodings of Theorem 4.13, the degree bound applies jointly to the original input and random bits; we obtain those encodings from the computationally private degree-three construction of [AIK06a], which uses the stronger easy PRG assumption.

6.3 Applications

We first use our multiparty encodings to reduce secure computation to parallel calls to a three-party functionality, and implement these calls to obtain three-round protocols in the two-round OT-hybrid model. We then establish privacy against weakly-active adversaries for the same protocols when the functionality has public outputs, before turning to non-interactive client–server computation.

6.3.1 Three-party functionalities and oblivious transfer

We consider the three-party functionality TOT, in which P_1 holds $(a, u) \in \mathbb{F}_2^2$, P_2 holds $(b, v) \in \mathbb{F}_2^2$, and P_3 has no input and receives $ab + u + v$, while the other two parties receive no output. We show that this fixed functionality is statistically complete under non-interactive reductions for functionalities with any number of parties, with privacy against every coalition.

To obtain these reductions, we apply the transformation of [AIKP22] to the fully private quadratic MPREs of Corollary 6.2. Each party prepares all its TOT inputs from its own input and randomness before receiving any answers. The calls are then evaluated in parallel, and the returned values allow each party to reconstruct its encoded output and apply its MPRE decoder. Different parties of a TOT call may be assigned to the same party of the original functionality, as allowed by Definition 6.4. The transformation has polynomial overhead and preserves both error bounds.

We also obtain three-round protocols by implementing these calls in the two-round OT-hybrid model. In this model, the parties communicate over synchronous private point-to-point channels, and an ideal OT receiving both parties' inputs in round i delivers its output at the end of round $i + 1$.

Corollary 6.8 (Statistical completeness and three-round computation)**.** *Let f be a deterministic k-party functionality represented by mod-2 branching programs of total size σ. For every security parameter $\lambda \geq 1$, f has a non-interactive reduction to parallel calls of TOT and a three-round protocol in the two-round OT-hybrid model, both with correctness and statistical privacy errors at most $2^{-\lambda}$ on every input. Privacy holds against every static, semi-honest coalition, and the complexity is polynomial in $\sigma + k + \lambda$.*

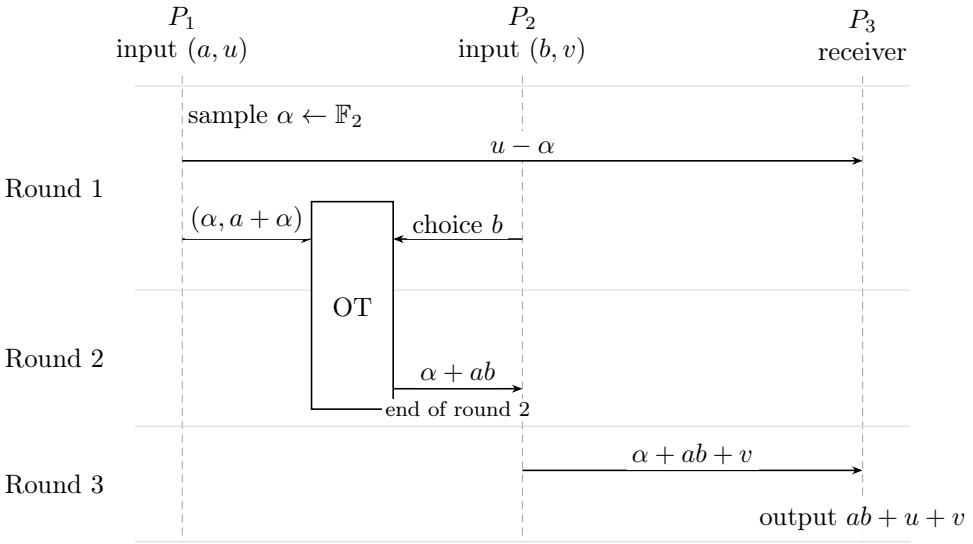


Figure 9: The three-round implementation of one TOT call in the two-round OT-hybrid model [AIKP22]. The parties submit their OT inputs in round one, P_2 receives the OT output at the end of round two, and its message to P_3 in round three completes the computation.

Implementing the calls with oblivious transfer. We use the perfectly correct and perfectly private implementation from the proof of [AIKP22, Theorem 10], which requires one OT call and two private messages. In round one, P_1 samples a uniform bit α, privately sends $u + \alpha$ to P_3, and supplies $(\alpha, a + \alpha)$ as its OT inputs, while P_2 supplies the choice bit b. At the end of round two, P_2 receives

the selected value $\alpha + ab$, to which it adds v before privately sending the result to P_3 in round three. The receiver adds its two messages to obtain

$$(u + \alpha) + (\alpha + ab + v) = ab + u + v.$$

We run these implementations in parallel for all calls, so the complete protocol takes three rounds and retains the MPRE's correctness and privacy bounds.

Applying the same transformations to Corollary 6.7 gives reductions and three-round protocols with complexity polynomial in circuit size, the number of parties and the security parameter. These constructions have computational privacy against every coalition and negligible correctness error, using only black-box access to a secure PRG.

6.3.2 Privacy against weakly-active adversaries

We now consider the same three-round protocols against the weakly-active adversaries of [AIKP22]. Such an adversary may replace an OT output only when both the sender and the receiver are corrupted, while the corrupted parties must sample their random tapes as prescribed and otherwise follow the protocol, using the replacement values in their subsequent computations.

The adversary fixes its coalition T and a vector a' of replacement bits before the execution. For every such choice, we require the coalition's view to be simulatable from $(T, a', x[T], f_T(x))$, while correctness is required only in passive executions, when all parties follow the protocol without replacing any OT outputs.

Corollary 6.9 (Statistical weakly-active privacy)**.** *Let f be a deterministic k-party functionality represented by mod-2 branching programs of total size σ. For every security parameter $\lambda \geq 1$, the three-round protocol of Corollary 6.8 has statistical privacy error at most $2^{-\lambda}$ against every weakly-active coalition and correctness error at most $2^{-\lambda}$ in passive executions, with both bounds holding on every input. Its complexity is polynomial in $\sigma + k + \lambda$.*

All OT inputs are submitted in round one, so replacements made when the outputs arrive at the end of round two can change the corrupted parties' outgoing messages only in the final round. The honest parties send their final-round messages before receiving these altered messages, and therefore send exactly what they would send in the corresponding passive execution. Given the coalition's passive view, which includes its inputs and complete random tapes, we can thus obtain its modified view by substituting a' and recomputing the corrupted parties' final-round messages and local outputs. Our simulator first samples a passive view from $(T, x[T], f_T(x))$ and then performs this computation. Applying the same deterministic map to the real and simulated passive views cannot increase their statistical distance, so the privacy error is unchanged, as is correctness in passive executions.

6.3.3 Non-interactive client–server computation

We now apply the polynomial-sharing construction of [ABT18] to our quadratic MPREs to obtain non-interactive client–server protocols using only private channels.

We first define the model precisely.

Definition 6.10 (Non-interactive client–server protocols)**.** Let $m, n \geq 1$ and let $F : X_1 \times \cdots \times X_m \to Y$ be a function between finite nonempty sets. Each input client C_i holds $x_i \in X_i$, while the receiver D has no input and is the only party with the prescribed output $F(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_m)$. The servers $S_1, \ldots, S_n$ have no inputs. We regard D as an additional client, so F defines an $(m+1)$-party

functionality. A protocol specifies finite nonempty tape sets $\mathcal{R}_P$ for all parties P, upload spaces $\mathcal{M}_{i,s}$ and $\mathcal{M}_{D,s}$, and reply spaces $\mathcal{Y}_s$. Each party independently samples a uniform tape $R_P \leftarrow \mathcal{R}_P$, which contains all its random choices. The protocol uses the following deterministic maps.

(i) *First round.* Each input client C_i and the receiver D send one message to each server S_s, given by

$$m_{i,s} = M_{i,s}(x_i, R_{C_i}) \in \mathcal{M}_{i,s}, \qquad m_{D,s} = M_{D,s}(R_D) \in \mathcal{M}_{D,s}.$$

These messages are sent simultaneously, using only the sender's input, if any, and its local tape.

(ii) *Second round.* Each server S_s uses its received uploads and local tape to compute

$$Y_s = H_s\big((m_{i,s})_{i=1}^m, m_{D,s}, R_{S_s}\big) \in \mathcal{Y}_s$$

and sends Y_s to D. After receiving all replies, D outputs

$$\mathrm{out}_D(\mathbf{x}) = \mathsf{Dec}(R_D, (Y_s)_{s=1}^n) \in Y \cup \{\perp\},$$

where $\perp \notin Y$ denotes failure.

All channels are private, and there is no other communication, setup, correlated initial randomness or ideal oblivious transfer. The local maps and decoder are otherwise unrestricted, and the receiver's tape includes any randomness used in its final computation. For $f^\star$, the input clients C_1, C_2, C_3 are A, B, C, respectively.

This is the designated-receiver case of the model of [AIKP22], with D treated as an additional inputless client. Although that model allows servers to reply to every client, we can delete the final replies to clients other than D without changing its output. Applying the same deletion to simulated views preserves privacy, so the lower bound below also applies to that more general interface. The client–server application of [BEGI26] also uses an inputless designated receiver, which it calls the *evaluator*.

For an integer $0 \le t < n$, we consider static, semi-honest coalitions in the class $\mathcal{A}_t$, consisting of any set of input clients, possibly the receiver D, and at most t servers.

Compiling encodings of low degree. We apply the client–server compiler of [ABT18, AIKP22] to an MPRE E of F over $\mathbb{F}_2$, with no internal randomness and with its encoded output delivered only to D. In this MPRE, each client first computes a vector of local values from its own input and randomness. We denote this computation by h_i: client C_i samples its MPRE tape τ_i and computes $w_i = h_i(x_i, \tau_i) \in \mathbb{F}_2^{a_i}$, without needing information from the other clients. The receiver also performs local preprocessing, even though it has no input, so we write $C_0 = D$, $x_0 = \perp$ and $w_0 = h_0(\perp, \tau_0)$.

The public polynomial map p combines these local vectors into the encoded output

$$z = p(w_0, \dots, w_m) \in \mathbb{F}_2^\ell,$$

which the receiver decodes using $\mathsf{Dec}_E(z, \tau_0)$. In the compiled protocol, each client shares the coordinates of its vector w_i, and the servers evaluate p on the shares they receive. We assume that the coordinate polynomials of p are explicitly given and have total degree at most $d \ge 1$, while the local preprocessing maps h_i have no degree restriction.

For our statistical MPRE, the map p comes directly from the ordinary quadratic encoder E_F of F. Each client, including the receiver, contributes an independent uniform share s_i of the encoder's

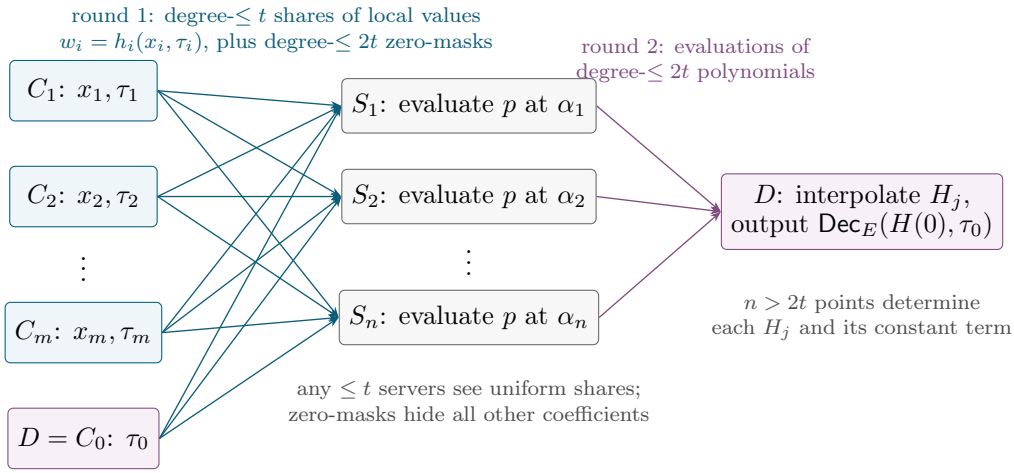


Figure 10: Construction 6.11 with a quadratic MPRE. The servers evaluate the public map on the uploaded shares and add the masks. From $n > 2t$ replies, D reconstructs the encoded output and applies the MPRE decoder.

random tape. We therefore take $w_i = (x_i, s_i)$ for each input client and $w_0 = s_0$ for the receiver, so that

$$p\big(s_0, (x_1, s_1), \ldots, (x_m, s_m)\big) = E_F\left(x_1, \ldots, x_m; \sum_{i=0}^{m} s_i\right).$$

Thus p adds the tape shares and evaluates the ordinary encoder on the clients' inputs using the resulting tape. Since this substitution is linear and E_F is quadratic, p has degree at most two.

Let $t \geq 0$ and $n > dt$ be two integers, $\mathbb{K} = \mathbb{F}_{2^\kappa}$, where $\kappa = \lceil \log_2(n+1) \rceil$, and distinct nonzero public points $\alpha_1, \ldots, \alpha_n \in \mathbb{K}$. The clients use Shamir sharing with polynomials of degree at most t [Sha79], and the servers evaluate the supplied coordinate polynomials of p over $\mathbb{K}$, keeping their coefficients in $\mathbb{F}_2 \subseteq \mathbb{K}$. Their replies are therefore evaluations of polynomials of degree at most dt, whose constant terms are the coordinates of z.
Since $n > dt$, the receiver can reconstruct these polynomials and recover z, but it also learns their other coefficients, which may reveal information about the clients' local values. We avoid this additional leakage by having each input client supply independent random polynomials of degree at most dt with constant term zero, which the servers add to their replies. These masks leave z unchanged, and the proof below shows that an honest client's contribution suffices to simulate the reply polynomials from the encoded output and the uploads known to corrupted servers.

We give the complete protocol in Construction 6.11, and Figure 10 illustrates the quadratic case, where reconstruction requires $n > 2t$.

Construction 6.11 (The compiled protocol Π_E)

PUBLIC PARAMETERS. $E, d, t, n, \mathbb{K}, \alpha_1, \ldots, \alpha_n$.

INPUTS. Client C_i holds x_i, for $i \in [m]$.

1. **Local computation and sampling.** Each C_i, for $0 \leq i \leq m$, independently samples its MPRE tape τ_i and computes $w_i = h_i(x_i, \tau_i)$. For each $k \in [a_i]$, it independently samples a

uniform polynomial $P_{i,k} \in \mathbb{K}[X]$ satisfying

$$\deg P_{i,k} \le t, \qquad P_{i,k}(0) = w_{i,k}.$$

Each input client C_i, for $i \in [m]$, also independently samples a uniform polynomial $Z_{i,j} \in \mathbb{K}[X]$ for each $j \in [\ell]$, satisfying

$$\deg Z_{i,j} \le dt, \qquad Z_{i,j}(0) = 0.$$

2. **First-round uploads.** Each C_i, including D, sends $(P_{i,k}(\alpha_s))_{k\in[a_i]}$ to every server S_s. Each input client also sends $(Z_{i,j}(\alpha_s))_{j\in[\ell]}$ to S_s.

3. **Second-round replies.** Each server S_s sends D the vector $(Y_{s,j})_{j\in[\ell]}$, where

$$Y_{s,j} = p_j\big((P_{i,k}(\alpha_s))_{0\le i\le m,\ k\in[a_i]}\big) + \sum_{i=1}^{m} Z_{i,j}(\alpha_s).$$

4. **Reconstruction and decoding.** For each $j \in [\ell]$, D interpolates a polynomial H_j of degree at most dt satisfying $H_j(\alpha_s) = Y_{s,j}$ for every $s \in [n]$. If no such polynomial exists, D returns $\perp$. Otherwise, it sets $\hat{z} = (H_j(0))_{j\in[\ell]}$ and returns $\mathsf{Dec}_E(\hat{z}, \tau_0)$ if $\hat{z} \in \mathbb{F}_2^{\ell}$, and $\perp$ otherwise.

Lemma 6.12 (The compiler)**.** *Let E be as above, with $d \ge 1$ and $n > dt$. Construction 6.11 gives a non-interactive client–server protocol with the same correctness error as E. It is perfectly private against every coalition $\mathcal{C} \in \mathcal{A}_t$ not containing D. When $D \in \mathcal{C}$, let $T = \mathcal{C} \cap \{C_0, \dots, C_m\}$. There is a randomized map $\Psi_{\mathcal{C}}$, depending only on the public parameters and the coalition, such that*

$$\mathrm{View}_{\mathcal{C}}(\mathbf{x}) \equiv \Psi_{\mathcal{C}}(V_T(\mathbf{x})), \qquad V_T(\mathbf{x}) = (\mathbf{x}[T], \tau[T], z).$$

Thus the protocol's privacy error against $\mathcal{C}$ is at most the MPRE's privacy error against T. The map $\Psi_{\mathcal{C}}$ is efficient whenever E is efficient, so computational privacy is also preserved. Each C_i, including D, sends a_i field elements to each server, and each input client sends ℓ additional field elements for the masks. Each server replies with ℓ field elements. The protocol and its simulators have polynomial overhead in $m + n$, the MPRE's input, tape and local-value lengths, and the descriptions and running times of its algorithms.

Proof. For each coordinate j, the server replies are evaluations of

$$H_j(X) = p_j\big((P_{i,k}(X))_{i,k}\big) + \sum_{i=1}^{m} Z_{i,j}(X).$$

This polynomial has degree at most dt and constant term z_j, because every masking polynomial has constant term zero. The receiver therefore reconstructs the exact encoded vector z and applies the original MPRE decoder with its tape τ_0, preserving the correctness error.

For privacy, fix a coalition $\mathcal{C}$ and let B index its corrupted servers. Since $|B| \le t \le dt$, the honest clients' uploads to B are jointly uniform even after fixing all inputs and MPRE tapes. If $D \notin \mathcal{C}$, these uploads are the only messages the coalition receives from honest parties. We can therefore simulate its view exactly by sampling the corrupted clients' complete tapes, computing their uploads, and sampling the honest clients' uploads to B independently and uniformly.

Suppose now that $D \in \mathcal{C}$. We construct $\Psi_{\mathcal{C}}$ from the MPRE view V_T by keeping the corrupted clients' MPRE tapes and sampling their additional sharing and masking randomness. If all input clients are corrupted, we know every input and MPRE tape, so we can generate the complete execution directly. Otherwise, we sample the honest clients' uploads to B independently and uniformly. Together with the corrupted clients' uploads, these determine every reply $Y_{s,j}$ from a corrupted server.

It remains to generate the replies from the honest servers. For each coordinate j, we independently sample a uniform polynomial from

$$\mathcal{H}_j = \{H \in \mathbb{K}[X] : \deg H \le dt,\ H(0) = z_j,\ H(\alpha_s) = Y_{s,j} \text{ for } s \in B\}.$$

The simulator knows all the values appearing in these constraints, and the set is nonempty because they specify at most $dt + 1$ evaluations at distinct points. Evaluating the sampled polynomials at the remaining server points gives the replies delivered to D.

To verify their distribution, fix V_T, the corrupted clients' sharing and masking randomness, and all uploads to B. We then fix all remaining randomness except the masks of one honest input client. For each coordinate j, that client's mask remains uniform among the polynomials of degree at most dt with constant term zero and its prescribed evaluations at B. Adding the fixed contribution of the other terms maps these possible masks bijectively onto $\mathcal{H}_j$. Since the masks for different coordinates are independent, the reply polynomials are independently uniform on their respective sets $\mathcal{H}_j$. These sets depend only on z and the uploads to B, so averaging over the additional fixing preserves this joint distribution.

This proves that $\Psi_{\mathcal{C}}$ generates the coalition's exact protocol view, including its complete random tapes. Applying $\Psi_{\mathcal{C}}$ to the MPRE simulator therefore preserves its statistical privacy error, and also preserves computational privacy when the algorithms are efficient. Sampling the constrained polynomials is efficient by linear algebra, while the communication bounds follow directly from the uploads and replies. □

The honest-majority threshold. We now apply Lemma 6.12 to our fully private quadratic MPREs, allowing the corrupted servers to collude with any set of clients, including the receiver.

Theorem 6.13 (Statistical client–server protocols under an honest majority)**.** *Let $0 \le t < n$ satisfy $n > 2t$, and let F be represented by mod-2 branching programs of total size σ, including its input and output lengths. For every security parameter $\lambda \ge 1$, F has a protocol in the model of Definition 6.10 with correctness error and statistical privacy error against $\mathcal{A}_t$ both at most $2^{-\lambda}$. Its construction and all protocol and simulation algorithms run in time polynomial in $m + n + \sigma + \lambda$, and its total communication is polynomial in the same parameters.*

Proof. We encode the entire output vector $F(\mathbf{x})$ using Theorem 4.12 at parameter λ, and then apply Proposition 6.1 to share the encoder's tape among the clients. Since only D receives an encoding, this conversion preserves the correctness bound $2^{-\lambda-2}$ and the privacy bound $\frac{3}{2}\,2^{-\lambda-2}$, regardless of the number of corrupted clients. Applying Lemma 6.12 with $d = 2$ gives the desired protocol whenever $n > 2t$, preserving both bounds with polynomial overhead. □

This threshold is optimal for general statistical computation, even when local computations are unrestricted. Lemma A.1 in Appendix A.1 shows that, when $n \le 2t$, every protocol for $f^\star$ with correctness error δ and statistical privacy error ε against $\mathcal{A}_t$ satisfies

$$\delta + 3\varepsilon \ge \tfrac{1}{2}.$$

Thus both errors cannot be negligible in this range. The proof applies the classical partition argument [CK91], with an explicit bound on the accumulated privacy error.

For functions represented by circuits, we instead apply the compiler to the computationally private MPREs of Corollary 6.7. These require only a secure PRG, whose evaluation takes place during local preprocessing.

Corollary 6.14 (Computational client–server protocols)**.** *Given black-box access to a secure pseudorandom generator, there is a construction that takes a circuit of size σ computing F, server parameters $0 \le t < n$ with $n > 2t$, and a security parameter 1^λ, and produces a protocol for F in the model of Definition 6.10 with correctness error at most $2^{-\lambda-2}$. The construction and all protocol and simulation algorithms run in time polynomial in $m + n + \sigma + \lambda$, and the total communication is polynomial in the same parameters.*

For every family of instances whose circuit sizes and numbers of clients and servers are polynomially bounded in λ, the resulting protocols are computationally private against $\mathcal{A}_t$, under the conventions of Definition 3.7.

Proof. We use Corollary 6.7 to obtain a computationally private 2MPRE of F that protects every coalition, uses no internal randomness and delivers its encoded output only to D. Lemma 6.12 then gives a client–server protocol with the same correctness bound $2^{-\lambda-2}$, preserving computational privacy for every polynomially bounded family. Both constructions have polynomial overhead, giving the stated complexity bounds. □

We may also obtain the PRG from black-box access to the algorithms of a semi-honest secure two-message OT protocol with negligible correctness error, using the standard reductions from OT to one-way functions and from one-way functions to PRGs [IL89, HILL99, ABG+20]. These algorithms are evaluated locally, so this use of OT does not change the client–server communication or require an OT execution between the parties.

Arbitrary private client outputs. To support several private outputs, we run independent copies of the single-receiver protocol in parallel, one for each client with a nonempty prescribed output vector. Each intended recipient performs the receiver's computation in its copy. If it also supplies an input, it performs the input client's computation with an independent tape, and both tapes are included in its view if it is corrupted. Each copy's simulator receives the corrupted clients' inputs and, when that copy's recipient is corrupted, the recipient's prescribed output.

For $r \ge 1$ nonempty output vectors, we use parameter $\lambda' = \lambda + \lceil \log_2 r \rceil$ in each copy. The simulation errors add over the copies, as do the correctness errors by a union bound, so the privacy error of the combined view and the probability that any recipient is incorrect are each at most $r2^{-\lambda'} \le 2^{-\lambda}$ in the statistical construction. Polynomially many copies of the computational protocol also preserve computational privacy, with the same correctness bound. If all prescribed outputs are empty, no communication is needed. The server threshold remains $n > 2t$, and the single-receiver lower bound shows that it is optimal for general statistical computation.

AI Methodologies Statement

The results were developed through an interaction with AI tools (Claude Opus 5.5 and GPT-6), with the authors providing the problem formulation, the central ideas, and the direction of the search. The tools contributed suggestions to the main construction and assisted with writing, figure formatting, and developing and checking some of the proofs, particularly the probability bounds

in Section 4.2. The authors have verified all arguments in full and take full responsibility for the manuscript and the correctness of its claims.

References

[ABG+20] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 71:1–71:18. LIPIcs, January 2020.

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 152–174. Springer, Cham, November 2018.

[AHI+17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 67, pages 7:1–7:31. LIPIcs, January 2017.

[AIK06a] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[AIK06b] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163. Springer, Berlin, Heidelberg, July 2010.

[AIK18] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Minimizing locality of one-way functions via semi-private randomized encodings. *Journal of Cryptology*, 31(1):1–22, January 2018.

[AIKP22] Benny Applebaum, Yuval Ishai, Or Karni, and Arpita Patra. Quadratic multiparty randomized encodings beyond honest majority and their applications. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 453–482. Springer, Cham, August 2022.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, Information Security and Cryptography, pages 1–44. Springer, 2017.

[BEGI26] Nir Bitansky, Saroja Erabelli, Rachit Garg, and Yuval Ishai. Shuffling is universal: Statistical additive randomized encodings for all functions. In *58th ACM STOC*. ACM Press, 2026.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[BIW10] Omer Barkol, Yuval Ishai, and Enav Weinreb. On d-multiplicative secret sharing. *Journal of Cryptology*, 23(4):580–593, October 2010.

[BL90] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, New York, August 1990.

[CK91] Benny Chor and Eyal Kushilevitz. A zero-one law for Boolean privacy. *SIAM Journal on Discrete Math.*, 4:36–47, 1991.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994.

[GIKM00] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000.

[HIKR23] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. Additive randomized encodings and their applications. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 203–235. Springer, Cham, August 2023.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Berlin, Heidelberg, July 2002.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Berlin, Heidelberg, May 2011.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS*, pages 230–235. IEEE Computer Society Press, October / November 1989.

[Ish13] Yuval Ishai. Randomization techniques for secure computation. In Manoj Prabhakaran and Amit Sahai, editors, *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 222–248. IOS Press, 2013.

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

A Proofs for the Client–Server Model

We show that the condition $n > 2t$ achieved by our client–server protocol is necessary for computing all functionalities with negligible correctness and statistical privacy errors in the model of Definition 6.10. When $n \le 2t$, we prove that every protocol for $f^\star$ satisfies $\delta + 3\varepsilon \ge \frac{1}{2}$, where δ and ε are its correctness and privacy errors. This bound holds even when the parties may perform arbitrary local computations.

A.1 Necessity of an honest server majority

The proof applies the standard partition argument [CK91] to two allowed coalitions and accounts explicitly for both correctness and privacy errors.

Lemma A.1 (A statistical lower bound for the server threshold)**.** *Suppose $0 \le t < n$ and $n \le 2t$. Any protocol for $f^\star$ in the model of Definition 6.10, with correctness error at most δ and statistical privacy error at most ε against $\mathcal{A}_t$, satisfies*

$$\delta + 3\varepsilon \ge \frac{1}{2}.$$

The bound holds without any restriction on the degree or computational complexity of the local maps.

Proof. We restrict the inputs to $x_A = (a, 0)$, $x_B = (b, 0)$ and $x_C = (1, 0)$, so that the prescribed output is ab. Since $n \le 2t$, we can partition the servers into sets U, V with $|U|, |V| \le t$ and form the allowed coalitions

$$\mathcal{L} = \{A, C, D\} \cup U, \qquad \mathcal{R} = \{B\} \cup V.$$

We obtain a two-party protocol by having the left party run all participants in $\mathcal{L}$ and the right party run all participants in $\mathcal{R}$, exchanging only the messages that cross the partition. The left party holds a and is prescribed the output ab, while the right party holds b and has no prescribed output. Each party's random tape consists of its members' tapes, so the two tapes remain independent and uniform.

Let T_{ab} denote the exchanged transcript on inputs (a, b), and let V^L_{ab} denote the complete left view. Whenever a coalition has the same inputs and prescribed output in two executions, both view distributions are within statistical distance ε of the same simulated distribution, and hence within 2ε of each other. Changing a while keeping b fixed leaves the right coalition's inputs and prescribed output unchanged. Similarly, when $a = 0$, changing b leaves the left coalition's inputs and prescribed output unchanged. Since either coalition's view determines the exchanged transcript, we obtain

$$\mathrm{SD}(T_{10}, T_{00}) \le 2\varepsilon, \qquad \mathrm{SD}(T_{00}, T_{01}) \le 2\varepsilon, \qquad \mathrm{SD}(T_{01}, T_{11}) \le 2\varepsilon.$$

The triangle inequality therefore gives $\mathrm{SD}(T_{10}, T_{11}) \le 6\varepsilon$.

We next compare the complete left views, which also include the left party's random tape. For a fixed transcript ω, let $K_L(a, \omega)$ be the set of left tapes that produce the outgoing messages specified by ω when the left party receives the incoming messages specified there, and define $K_R(b, \omega)$ analogously. A pair of tapes produces ω exactly when it belongs to $K_L(a, \omega) \times K_R(b, \omega)$. Since the tapes are independent and uniform, conditioning on a transcript of positive probability makes the left tape uniform on $K_L(a, \omega)$, whose definition does not depend on b.

With $a = 1$ fixed, we can therefore generate the complete left view from either transcript distribution by the same randomized procedure, sampling a uniform tape from $K_L(1, \omega)$ and reconstructing the left party's local execution. This procedure need not be efficient and can be

defined arbitrarily on transcripts that occur in neither execution. Applying the same randomized procedure cannot increase statistical distance, so

$$\mathrm{SD}(V_{10}^L, V_{11}^L) \le \mathrm{SD}(T_{10}, T_{11}) \le 6\varepsilon.$$

Finally, let O_{ab} denote the receiver's output. Correctness gives $\Pr[O_{10} = 1] \le \delta$ and $\Pr[O_{11} = 1] \ge 1 - \delta$, where a failure symbol counts as an error. Since the receiver belongs to the left group, its output is determined by the complete left view. Consequently,

$$1 - 2\delta \le \Pr[O_{11} = 1] - \Pr[O_{10} = 1] \le \mathrm{SD}(V_{11}^L, V_{10}^L) \le 6\varepsilon.$$

Rearranging gives $\delta + 3\varepsilon \ge \frac{1}{2}$. □

B Details of Non-Interactive Reductions

We prove Lemma 6.6, following the composition argument of [ABT18, Lemmas 1–2] and tracking the correctness and privacy errors separately.

Proof. We construct the encoding by letting each party prepare its call inputs as prescribed by Π and then compute the local values required by the copies of E. We use fresh independent randomness for every copy, including any internal randomness, independently of the reduction tapes. When one protocol party acts for several parties in the same call, we give them separate independent local tapes, so the copy has exactly the randomness distribution required by E. We then form the public map by concatenating the outputs of the copies' quadratic maps. Since the computations needed to prepare their inputs are included in local preprocessing, the resulting encoding has effective degree at most two and uses no internal randomness when E has none.

To decode, we recover the bit from each call for which the party is the receiver and then apply its output map from Π, using its original input and reduction tape. We supply the prescribed empty call outputs directly, since their values are already known, and return failure if a call decoder reports a failure symbol. To bound the correctness error, we fix the input and all reduction tapes, which fixes every call input while leaving the encoding randomness independent and uniform. Each party needs at most M decoded bits, each with error probability at most δ_1, so a union bound gives probability at most $M\delta_1$ that any required bit is decoded incorrectly. Whenever all these bits are correct, we obtain exactly the party's output in Π, and therefore averaging over the reduction tapes and adding the reduction's own error gives $\delta_0 + M\delta_1$.

For privacy, we describe a procedure Ψ_T that generates a simulated encoding view from the coalition's reduction view. In call j, the coalition acts for the parties in $T_j = \pi_j^{-1}(T)$, whose call inputs we can compute from the coalition's original inputs and reduction tapes. Their prescribed call outputs are also included in the reduction view, so we have all the information needed to run the simulator of E for T_j. We may do this even when T_j contains all four parties, because E is fully private. We run these call simulations independently and combine the new local tapes and encoded outputs they jointly produce with the coalition's original inputs and reduction tapes. This defines Ψ_T on valid reduction views; on malformed inputs, we let it return a fixed default view.

To compare this procedure with the real encoding, we again fix all reduction tapes, so that the call inputs are fixed and the copies use independent randomness. We can then replace the call views one at a time by their simulations, changing the distribution by at most ε_1 at each replacement. After averaging over the reduction tapes, the resulting distribution is $\Psi_T(\mathrm{View}_T^{\Pi}(\mathbf{x}))$, within statistical distance $M\varepsilon_1$ of the real encoding view. We finally replace the reduction view by

$\mathsf{Sim}_\Pi(T, \mathbf{x}[T], f_T(\mathbf{x}))$, which adds at most ε_0 because applying Ψ_T cannot increase statistical distance. Our simulator therefore runs Sim_Π followed by Ψ_T, with total privacy error at most $\varepsilon_0 + M\varepsilon_1$.

For computationally private reductions, we use the same simulator and apply computational indistinguishability in the last comparison, since Ψ_T is efficient and the preceding statistical error $M\varepsilon_1$ is negligible by assumption. The construction, decoding and simulation use the corresponding algorithms of Π and M copies of E, giving polynomial overhead. When $M = 0$, we simply use an empty public output and let each party apply its local output map from Π as its decoder. $\square$