On the Power of Slicing and Dicing: Linear Garbling Beyond Two-Input Gates

Tianren Liu and Luojian Wei

Peking University

Abstract. Communication is a central efficiency bottleneck in garbled circuits. Rosulek and Roy (CRYPTO 2021) introduced *slicing* and *dicing*, obtaining a garbling scheme in which XOR gates require no communication and each two-input AND gate costs $1.5\lambda + 5$ bits, where λ is the security parameter.

In this work, we investigate the power of these techniques for garbling gates of larger fan-in. We develop a linear-algebraic framework that yields concrete constructions and asymptotic bounds. For three- and four-input gates, we obtain constructions with communication costs $7\lambda/3 + 35$ and $15\lambda/4 + O(1)$ bits, respectively. For arbitrary fan-in n, we give a construction for any single-output gate with communication cost bounded by $22 \cdot 2^n\lambda/n + 2^n$ bits.

Our constructions are compatible with free-XOR, and all hash queries within each gate are nonadaptive. We prove security of the framework in the random-oracle model and under a circular correlation robustness (CCR) condition for any number of slices.

We also evaluate simple uses of the three-input construction against the best two-input circuits collected in our experiments, finding evidence of communication savings on control circuits.

1 Introduction

Garbled circuits (GC), introduced by Yao [22], provide an efficient approach to constant-round secure two-party computation. They use efficient symmetric-key operations, but transmitting the garbled circuit remains a major practical bottleneck.

Considerable effort has been devoted to reducing the concrete size of GC [5, 10,14,15,17,19,20,25]. The classical construction follows the *gate-by-gate* framework: the computation is represented by a fan-in-2 Boolean circuit. The garbler samples a key for each input wire, then garbles the circuit gate by gate. For each gate, based on the keys of its input wires, the garbler generates a *table* associated with that gate and a key for its output wire. The garbled circuit is the collection of all tables. In this framework, the optimization goal is to minimize the table size of each gate.

The free-XOR technique allows XOR and NOT gates to be garbled without communication [15]. For AND gates, the half-gates construction reduces the communication cost to 2λ bits [25], where λ is the security parameter. Rosulek

and Roy [20] further reduce this cost to $1.5\lambda + 5$ bits through *slicing* and *dicing*. Their construction organizes linear computations over fragments of wire labels, enabling finer-grained control over the amount of information transmitted for each gate. Their scheme is secure under a circular correlation-robustness (CCR) assumption.

Several works establish lower bounds with leading term $3\lambda/2$ for two-input AND gates in models that accommodate slicing and dicing [3, 9, 12, 21]. Under their respective model restrictions, these bounds match RR21 up to constant additive overhead. This motivates a complementary direction: understanding the power of slicing and dicing when the basic unit of computation is a gate with more than two inputs. This leads to two questions:

> *How far can slicing and dicing be extended beyond two-input gates? When does representing a circuit with higher-fan-in gates reduce its garbled size?*

Direct garbling of larger gates has also been studied. For arbitrary n-input, single-output Boolean gates, Heath, Kolesnikov, and Ng [11] give a construction with communication cost $2^n + O(n\lambda)$ bits. For three-input gates, however, its cost of $5\lambda + 8$ bits exceeds the $3\lambda + 10$-bit worst-case cost of an RR21 decomposition into at most two AND gates and free XOR and NOT gates [7]. We therefore focus on the following refined question:

> *Can we garble arbitrary three-input gates using fewer than* $3\lambda + 10$ *bits of communication?*

Our circuit experiments provide a concrete motivation for this question. In the binary expansions of our selected mixed control circuits, 40.31% of the AND gates in aggregate, and up to 81.36% in one circuit, form pairs that can be replaced by three-input gates. These expansions need not coincide with the best collected binary references, against which we separately measure communication savings. Section 6.2 defines this proportion and reports the comparisons.

Our Results. We develop a linear-algebraic framework for garbling larger-fan-in gates (Section 4). The framework expresses correctness and security through three conditions on a system of linear equations. We prove that these conditions suffice for security in the random-oracle (RO) model, reducing the verification of each construction to checking explicit algebraic conditions. Section 4.5 formulates the corresponding CCR condition for s slices, under which we also prove obliviousness, privacy, and authenticity. Both analyses use the same garbling equations.

We also prove a communication lower bound within the linear model defined by these conditions. For every $n \geq 2$, any construction in this model supporting arbitrary n-input, single-output Boolean gates requires at least

$$\max\{(2^n - n - 1)\lambda/(n-1), (2^n - 1)\lambda/n\} \quad \text{bits.}$$

For $3 \le n \le 7$, we obtain constructions attaining the lower bounds in the coefficient of λ. In particular, three- and four-input gates cost $7\lambda/3 + O(1)$ and $15\lambda/4 + O(1)$ bits, respectively. These constructions are specified by explicit matrices, and their correctness and security conditions are verified by computer programs.

For arbitrary fan-in $n \ge 2$, we give a general construction with communication cost bounded by $22 \cdot 2^n\lambda/n + 2^n$. Thus the upper and lower bounds match up to a constant factor in the term proportional to λ, whose optimal order within our framework is $\Theta(2^n/n)$. Our general construction uses only nonadaptive oracle calls within each gate, unlike the construction presented by Heath, Kolesnikov, and Ng [11]. The general construction follows a separate route from the finite instances and is supported by an existence proof for every fan-in.

Three-input gates provide the smallest setting in which to examine the concrete efficiency improvement of this extension. Every three-input Boolean function can be computed using at most two two-input AND gates, together with XOR and NOT gates [7]. Garbling such a decomposition with the scheme of Rosulek and Roy therefore incurs a worst-case communication cost of $3\lambda + 10$ bits. Before slice padding, our three-input component costs $7\lambda/3+35$ bits, saving $2\lambda/3 - 25$ bits. The evaluations below include padding, using the exact count $7\lceil\lambda/3\rceil + 35$.

We therefore complement these constructions with communication-cost evaluations against the best two-input candidates collected for each function. We evaluate existing three-input networks and simple local replacements, selecting candidates according to their communication cost. For ten control-circuit benchmarks, the selected circuits reduce aggregate estimated communication by 5.92%, 7.30%, and 8.18% at $\lambda = 128, 256, 512$, respectively, relative to the best verified two-input references in our pool. Gains in the other categories are modest. These results provide evidence that three-input garbling can be useful for circuit-level optimization. A more systematic synthesis procedure may find further opportunities; our experiments do not establish the best achievable mixed circuits.

Technical Overview. We start from the linear evaluation used in RR21. For an n-input gate, divide each wire label into s equal slices and let the table contain r slices, costing $r\lambda/s$ bits. The evaluator recovers each slice of the output label by linearly combining input-label slices, table entries, and hash outputs of the same slice length.

The coefficients depend on public *color bits*: wire values masked by secret permutation bits. Writing the gate function in these coordinates gives its *permuted truth table*. The scheme fixes how table entries and hashes are combined at each color; the coefficients multiplying input labels, called *control coefficients*, are chosen using the permuted truth table. Substituting the free-XOR relations between wire labels gives a linear system, the *garbling equation*, which must hold at every input color. Section 4 formalizes the following three conditions and proves that they suffice for correctness and security.

Correctness (C). For every permuted gate function and every choice of valid input labels and hash values, the garbler can choose control coefficients, output labels, and table entries satisfying all the equations. A single garbled table must therefore support correct evaluation at every input.

Security I (S1). The control coefficients are sampled uniformly from the solutions allowed by C. The evaluator learns only their values at its own input colors. S1 requires these local values to be jointly uniform, independently of the permuted truth table, so they reveal no information about the secret permutation.

Security II (S2). Hashes on labels unavailable to the evaluator must independently mask all r table slices. Equivalently, the matrix mapping these unknown hash values to the table must have full row rank. This makes the table uniform after fixing the known hashes and the other terms in the garbling equation, enabling the simulation argument.

These conditions guide two construction methods. For small fan-in, we build table-coefficient columns from earlier columns, forming a directed acyclic graph. Each column records how one table slice contributes at different input colors. Multiplying combinations of earlier columns by linear functions of the colors produces higher-degree terms. As in RR21, each column receives a complementary hash pair, with one hash unavailable at each input color. The acyclic structure arranges these unknown hashes into a triangular masking matrix. We randomly choose coefficients and retain candidates that pass the correctness and security checks. Section 5.1 gives the construction, search procedure, and verified instances.

For arbitrary fan-in, we instead select products of color bits, called Boolean monomials, and use the same selection for every output slice. The aim is to choose a small collection from which every nonconstant monomial can be obtained by multiplying a selected monomial by one additional color bit. For example, selecting xy also gives access to xyz by multiplication by z. This turns the design into a covering problem on subsets of the input variables. Known covering results yield a collection of size $O(2^n/n)$ and hence our general communication bound. Section 5.2 develops this construction and verifies its correctness and security conditions.

Related Work. Several works formalize the restrictions under which communication lower bounds for garbling can be proved. The linear model of Zahur, Rosulek, and Evans [25] yields a 2λ lower bound for two-input AND gates; the slicing technique of Rosulek and Roy [20] goes beyond that model. Several subsequent works establish a leading lower bound of $3\lambda/2$ for two-input AND gates in models that allow slicing, under their respective restrictions [3, 9, 12, 21].

Extending sliced garbling to three inputs has required particular care with privacy. Ashur, Hazay, and Satish reported early attempts at CFAIL [1]; their later work [2] proposed a construction for $a(b + c)$ whose control information was shown to leak secret permutation bits by Fan, Lu, and Zhou [9] and by Kim [13]. The algebraic analyses of Ashur et al. and Kim also identify a degree-two obstruction when the coefficient space is restricted to affine functions of

the public input colors. Our construction allows these coefficients to have higher degree in the colors, while remaining linear in label slices, hash outputs, and table entries. Thus linearity of the label operations is distinct from affine dependence on the colors. Building on the existing garbling-equation viewpoint, we give simultaneous sufficient matrix conditions for correctness and security in the RO model and use them to construct and verify higher-fan-in instances.

Previous models also address higher fan-in directly. Fan, Lu, and Zhou prove an $n\lambda$ lower bound for an n-input AND gate in their single-base, column-wise model [9]; this bound does not cover their more general variants with correlations between multiple bases. Xu, Hu, and Xu prove a $7\lambda/4$ lower bound in their fan-in-three bitwise model with quasi-free-XOR [21]. Their construction attaining this bound applies to truth tables of even parity, meaning an even number of entries equal to one, and does not handle the three-input AND gate. Our three-input construction supports every truth table, including odd-parity gates, at $7\lambda/3 + O(1)$ bits. These comparisons concern different construction families: our own lower bound is stated for the matrix conditions developed here, rather than for all linear or bitwise garbling schemes.

2 Preliminaries

2.1 Notations

Throughout this paper:

- $[N]$ denotes the set $\{1, 2, \ldots, N\}$.
- λ denotes the security parameter.
- $\mathbb{F}_{2^n}$ denotes the finite field of size 2^n.
- "$\oplus$" and "$+$" denote the XOR operation between bit strings (viewed as elements of $\mathbb{F}_{2^\lambda}$ or $\mathbb{F}_{2^{\lambda/2}}$) and Boolean variables in $\mathbb{F}_2$, respectively. Since these are all characteristic-2 fields, "$\oplus$" and "$+$" are mostly interchangeable.
- $\Delta \in \mathbb{F}_{2^\lambda}$ denotes the global offset in the free-XOR setting.
- Subscripts L, R denote the first and second halves of a λ-bit value. For example, Δ is a dimension-2 vector $\Delta = (\Delta_L, \Delta_R)$.
- H denotes the hash function.

For most of the paper, we focus on garbling a single gate g and use the following notation:

- x, y, z, w denote color bits.
- $\alpha, \beta, \gamma, \delta$ denote permute bits.
- A^x, B^y, C^z, D^w denote labels with color bits x, y, z, w, respectively.
- x, α, A are associated with the first input wire; y, β, B with the second. For a 2-input gate, z, γ, C are associated with the output wire. For a 3-input gate, z, γ, C are associated with the third input wire, and w, δ, D with the output wire.

– $\tilde{g}$ denotes the permuted gate function, defined according to fan-in as

$$\tilde{g}(x,y) = \gamma + g(x+\alpha, y+\beta),$$
$$\tilde{g}(x,y,z) = \delta + g(x+\alpha, y+\beta, z+\gamma).$$

– $\mathcal{T}(\tilde{g})$ denotes the truth-table column vector of $\tilde{g}$, indexed by input colors in lexicographic order.
– Boldface uppercase letters such as $\mathbf{V}$, $\mathbf{M}$, and $\mathbf{R}$ represent matrices over $\mathbb{F}_2$, used as coefficient matrices during the garbling process. Each such matrix stacks four or eight equally sized blocks, one for each input color combination in lexicographic order. During evaluation, only the block indexed by the evaluator's colors, such as $\mathbf{V}(x,y)$ or $\mathbf{V}(x,y,z)$, is used. For a 2-input gate, for example,

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}(0,0) \\ \mathbf{V}(0,1) \\ \mathbf{V}(1,0) \\ \mathbf{V}(1,1) \end{bmatrix}.$$

– $\mathbf{A} \otimes \mathbf{B}$ denotes the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$.
– colspace($\mathbf{V}$) denotes the span of the column vectors of $\mathbf{V}$.
– $\mathbf{G} = (G_0, G_1, \dots)$ denotes the table (of ciphertexts) associated with a gate.

2.2 Circuits

A circuit C is formalized as a DAG consisting of $|C|$ nodes, each representing a wire, indexed from 1 to $|C|$ in topological order. During evaluation, let x_i denote the semantic value of the i-th wire. A subset of the wires $\mathsf{in}(C) \subseteq [|C|]$ are input wires (by default $\mathsf{in}(C) = [|\mathsf{in}(C)|]$ consists of the wires with the smallest indices); their values are directly determined by the input to the circuit. Every non-input wire $i \notin \mathsf{in}(C)$ is the output of some gate. Let g_i be the gate function and let $\mathsf{in}(i) = \{j_{i,1}, \dots\} \subseteq [i-1]$ denote its incoming wires; the value x_i is then determined by $x_i = g_i(x_{j_{i,1}}, \dots)$.

2.3 Garbling Scheme

We follow the definition of garbling scheme from [6].

Definition 1 (Garbling Scheme). *A garbling scheme consists of four algorithms* $(\mathsf{Garb}, \mathsf{Eval}, \mathsf{Enc}, \mathsf{Dec})$ *for circuits* $C : \{0,1\}^n \to \{0,1\}^m$*:*

– $\mathsf{Garb}(1^\lambda, C) \to (\tilde{C}, e, d)$ *garbles circuits. It takes as input the security parameter* λ *and a circuit description* C*, and returns a garbled circuit* $\tilde{C}$ *along with* e, d *that respectively contain information for encoding and decoding.*
– $\mathsf{Enc}(e, x) \to X$ *encodes the party inputs. It takes as input the encoding string* e *and a cleartext input* $x \in \{0,1\}^n$*, and returns input wire labels* X*.*
– $\mathsf{Eval}(\tilde{C}, X) \to Y$ *evaluates the GC. It takes as input the garbled circuit* $\tilde{C}$ *and input wire labels* X*, and returns output wire labels* Y*.*

– $\mathsf{Dec}(d, Y) \to y$ *decodes circuit outputs. It takes as input the decoding string* d *and output wire labels* Y*, and returns* $y \in \{0,1\}^m$ *or* $\perp$ *indicating failure.*

satisfying the following correctness condition for all circuits C *and inputs* $x \in \{0,1\}^n$*:*

$$\mathsf{Dec}(d, \mathsf{Eval}(\tilde{C}, \mathsf{Enc}(e, x))) = C(x), \text{ where } (\tilde{C}, e, d) \leftarrow \mathsf{Garb}(1^\lambda, C).$$

When discussing security, we focus on *obliviousness* as defined in [6], which requires that $(\tilde{C}, X)$ reveals no private information of the garbler and can be simulated. Although [6] also defines *privacy* and *authenticity*, which are incomparable with obliviousness, for most linear garbling schemes—including ours—these two properties follow easily from obliviousness. We prove that our scheme satisfies all three properties in Appendix A.

3 Previous Works

This section provides a concise review of existing 2-input gate garbling schemes. The aim is to build an intuitive understanding of our 3-input gate garbling framework by showing how it evolved from prior constructions.

We consider garbling a 2-input gate g under the free-XOR setting [15]. The two input wires carry labels $(A^0, A^1 := A^0 \oplus \Delta)$ and $(B^0, B^1 := B^0 \oplus \Delta)$. The evaluator learns one label per wire; its least significant bit—the superscript—serves as a *color bit*, so the evaluator also learns x, y when it holds A^x, B^y. These labels correspond to real wire values $x + \alpha, y + \beta$, where α, β are *permute bits* chosen secretly by the garbler. Let $(C^0, C^0 \oplus \Delta)$ and γ be the labels and permute bit of the output wire; the evaluator computes the output label C^z corresponding to the real wire value $z + \gamma = g(x + \alpha, y + \beta)$.

3.1 Unoptimized Free-XOR

We start with the original free-XOR linear garbling [15]. The table consists of 4 ciphertexts $\mathbf{G} = (G_{0,0}, G_{0,1}, G_{1,0}, G_{1,1})$; the evaluator computes the next label from the (x, y)-th entry $G_{x,y}$ by[1]

$$C^z = G_{x,y} \oplus H(A^x, B^y).$$

Requiring this equation to hold for every color pair (x, y) determines the table $\mathbf{G}$ via $G_{x,y} = C^0 \oplus (\gamma + g(x + \alpha, y + \beta))\Delta \oplus H(A^x, B^y)$.

Hence the garbler, given A^0, B^0 and Δ, samples a random C^0 and generates $\mathbf{G} = (G_{x,y})$ following the above constraints. Equivalently, the garbler samples a

[1] To avoid collision, the gate index should be passed to the hash function as well. We omit the gate indexes when garbling a single gate.

random solution of the following linear equation.

$$\begin{bmatrix}1&1&0&0&0\\1&0&1&0&0\\1&0&0&1&0\\1&0&0&0&1\end{bmatrix}\begin{bmatrix}C^0\\G_{0,0}\\G_{0,1}\\G_{1,0}\\G_{1,1}\end{bmatrix}=\begin{bmatrix}H(A^0,B^0)\\H(A^0,B^1)\\H(A^1,B^0)\\H(A^1,B^1)\end{bmatrix}\oplus\underbrace{\begin{bmatrix}\gamma+&g(\alpha,\beta)\\\gamma+&g(\alpha,1+\beta)\\\gamma+&g(1+\alpha,\beta)\\\gamma+&g(1+\alpha,1+\beta)\end{bmatrix}}_{\mathcal{T}(\tilde{g})}\Delta,$$

where $\mathcal{T}(\tilde{g})$ denotes the truth table of the permuted function $\tilde{g}(x,y) := \gamma + g(x+\alpha, y+\beta)$.

3.2 Free-XOR (with Row Reduction)

In the scheme above, $\mathbf{G}$ must accommodate the 4-dimensional freedom of an arbitrary gate function $\tilde{g}$; the dimension of $\mathbf{G}$ can therefore be no smaller. The row reduction technique [17] exploits the fact that C^0 supplies one dimension of freedom, saving one ciphertext from $\mathbf{G}$. Applying this technique makes C^0 dependent on $A^0, B^0, \Delta, \tilde{g}$, and H.

Informally, this is equivalent to "forcing $G_{0,0}$ to be 0." The garbler solves for $(C^0, G_{0,1}, G_{1,0}, G_{1,1})$ from the following equation.

$$\underbrace{\begin{bmatrix}1&0&0&0\\1&1&0&0\\1&0&1&0\\1&0&0&1\end{bmatrix}}_{\mathbf{V}}\begin{bmatrix}C^0\\G_{0,1}\\G_{1,0}\\G_{1,1}\end{bmatrix}=\begin{bmatrix}H(A^0,B^0)\\H(A^0,B^1)\\H(A^1,B^0)\\H(A^1,B^1)\end{bmatrix}\oplus\mathcal{T}(\tilde{g})\Delta$$

A solution is guaranteed since $\mathbf{V}$ is invertible.

3.3 Half-Gates

The half-gates garbling scheme [25] reduces the table to just two ciphertexts, G_0 and G_1. Rather than seeking additional sources of freedom, this technique reduces the dimension of $\tilde{g}$ itself by incorporating the input labels A^x, B^y into the linear system. For a permuted 2-input AND gate $\tilde{g}(x,y) = \gamma + (x+\alpha)(y+\beta) = xy + \alpha y + \beta x + \alpha\beta + \gamma$, the authors eliminate the quadratic term $xy\Delta$ from $\tilde{g}(x,y)\Delta$ by adding $yA^x = yA^0 \oplus xy\Delta$. The remaining expression $\tilde{g}(x,y)\Delta \oplus yA^x$ is linear in x, y, and thus can be represented using fewer dimensions.

Concretely, the evaluator computes the output label as

$$C^z = xG_0 \oplus yG_1 \oplus H(A^x) \oplus H(B^y) \oplus yA^x.$$

To guarantee correctness, C^0, G_0, G_1 must satisfy

$$\underbrace{\begin{bmatrix}1&0&0\\1&0&1\\1&1&0\\1&1&1\end{bmatrix}}_{\mathbf{V}}\begin{bmatrix}C^0\\G_0\\G_1\end{bmatrix}=\underbrace{\begin{bmatrix}1&0&1&0\\1&0&0&1\\0&1&1&0\\0&1&0&1\end{bmatrix}}_{\mathbf{M}}\begin{bmatrix}H(A^0)\\H(A^1)\\H(B^0)\\H(B^1)\end{bmatrix}\oplus\begin{bmatrix}0&0\\1&0\\0&0\\1&1\end{bmatrix}\begin{bmatrix}A^0\\\Delta\end{bmatrix}\oplus\mathcal{T}(\tilde{g})\Delta,\qquad(1)$$

where $\tilde{g}$ is a permuted AND gate. Equivalently,

$$C^0 \oplus xG_0 \oplus yG_1 = \underbrace{\begin{bmatrix} x+1 & x & y+1 & y \end{bmatrix}}_{\mathbf{M}(x,y)} \begin{bmatrix} H(A^0) \\ H(A^1) \\ H(B^0) \\ H(B^1) \end{bmatrix} \oplus y(A^0 + x\Delta) \oplus \tilde{g}(x,y)\Delta. \quad (2)$$

The right-hand side is linear in (x, y) thus can be expressed using only C^0, G_0, and G_1.

3.4 Three-Halves

The state-of-the-art three-halves garbling scheme [20] requires only $1.5\lambda + 5$ bits of communication per AND gate. It extends the idea of injecting input labels into the linear system to simplify $\tilde{g}$ and reduce its effective dimension. Unlike the half-gates scheme, where yA^x always eliminates the quadratic term $xy\Delta$, here the simplification is gate-dependent: the garbler chooses the coefficients of the input labels in the linear system adaptively. To achieve finer control and to preserve security despite this gate dependence, RR21 introduces two key techniques, *slicing* and *dicing*.

Slicing means that every label is evenly divided into two halves. For example, $A^0 = (A^0_L, A^0_R), B^y = (B^y_L, B^y_R), \Delta = (\Delta_L, \Delta_R)$. These are now viewed as vectors in $\mathbb{F}^2_{2^{\lambda/2}}$. The table consists of three $\lambda/2$-bit ciphertexts $\mathbf{G} = (G_0, G_1, G_2)$. *Dicing* means that the coefficients of the input labels are drawn uniformly at random from the solution space, and only the marginal values at the evaluator's color bits are revealed; this leaks no information about the permute bits.

The evaluator computes the output label $C^z = (C^z_L, C^z_R)$ by

$$\begin{aligned} C^z = \mathbf{V}^*(x,y)\mathbf{G} \oplus \begin{bmatrix} H(A^x) \oplus H(A^x \oplus B^y) \\ H(B^y) \oplus H(A^x \oplus B^y) \end{bmatrix} \\ \oplus \mathbf{R}_A(x,y) \cdot A^x \oplus \mathbf{R}_B(x,y) \cdot B^y. \end{aligned}$$

where $\mathbf{V}^*(x,y)$, $\mathbf{R}_A(x,y)$, and $\mathbf{R}_B(x,y)$ are 2-row matrices (one row per slice).

To express correctness at all color pairs, let $\mathbf{V}(x,y) = [\mathbf{I}\ \mathbf{V}^*(x,y)]$ denote the concatenation of the identity matrix and $\mathbf{V}^*(x,y)$. The garbler must find $(C^0_L, C^0_R, G_0, G_1, G_2)$ satisfying for all $(x,y) \in \{0,1\}^2$:

$$\begin{aligned} \mathbf{V}(x,y) \begin{bmatrix} C^0 \\ \mathbf{G} \end{bmatrix} = \begin{bmatrix} H(A^x) \oplus H(A^x \oplus B^y) \\ H(B^y) \oplus H(A^x \oplus B^y) \end{bmatrix} \\ \oplus \mathbf{R}_A(x,y) \cdot A^0 \oplus \mathbf{R}_B(x,y) \cdot B^0 \\ \oplus (x\mathbf{R}_A(x,y) + y\mathbf{R}_B(x,y))\,\Delta \oplus \tilde{g}(x,y)\Delta. \end{aligned}$$

This set of constraints can be equivalently expressed as the following equation. Here we instantiate $\mathbf{V}$ with the actual coefficients used in [20].

$$\underbrace{\left[\begin{array}{c|c|c|c|c|c} 1&0&0&0&0&0\\ \hline 0&1&0&0&0&0\\ \hline 1&0&0&0&0&1\\ \hline 0&1&0&1&1&1\\ \hline 1&0&1&0&1&\\ \hline 0&1&0&0&1&\\ \hline 1&0&1&0&0&\\ \hline 0&1&0&1&0&\end{array}\right]}_{\mathbf{V}} \begin{bmatrix} C_L^0\\ C_R^0\\ G_0\\ G_1\\ G_2 \end{bmatrix} = \underbrace{\left[\begin{array}{c|c|c|c|c|c} 1&0&0&0&1&0\\ \hline 0&0&1&0&1&0\\ \hline 1&0&0&0&0&1\\ \hline 0&0&0&1&0&1\\ \hline 0&1&0&0&0&1\\ \hline 0&0&1&0&0&1\\ \hline 0&1&0&0&1&0\\ \hline 0&0&0&1&1&0\end{array}\right]}_{\mathbf{M}} \begin{bmatrix} H(A^0)\\ H(A^1)\\ H(B^0)\\ H(B^1)\\ H(A^0\oplus B^0)\\ H(A^0\oplus B^1) \end{bmatrix} \begin{array}{l} \oplus \mathbf{R}_A \cdot A^0 \oplus \mathbf{R}_B \cdot B^0 \\ \quad \oplus \left(\mathbf{R}_\Delta + \mathcal{T}(\tilde{g}) \otimes \mathbf{I}\right) \Delta \end{array} \tag{3}$$

where $\mathbf{R}_\Delta$ is defined as $\mathbf{R}_\Delta(x,y) = x\mathbf{R}_A(x,y) + y\mathbf{R}_B(x,y)$.

To ensure solvability of eq. (3), the coefficients $\mathbf{R}_A, \mathbf{R}_B$ must be chosen so that $\mathbf{R}_A, \mathbf{R}_B$ and $\mathbf{R}_\Delta + \mathcal{T}(\tilde{g}) \otimes \mathbf{I}$ lie in the column span of $\mathbf{V}$ (one readily verifies that $\mathbf{M}$ is already in colspace($\mathbf{V}$)). This introduces a vulnerability: $\mathbf{R}_A, \mathbf{R}_B$ depend on the permuted gate $\tilde{g}$, and may therefore leak information about the permute bits.

Such leakage is prevented by *dicing*. The garbler samples $\mathbf{R}_A, \mathbf{R}_B$ jointly at random under the constraints. During evaluation, only $\mathbf{R}_A(x,y)$ and $\mathbf{R}_B(x,y)$ are revealed. Their joint marginal distribution must be independent of the permute bits. Under this condition, the coefficients simplify $\tilde{g}$ obliviously and preserve security.

3.5 Remark

The central challenge in this line of work is to capture the freedom of a gate function within a linear system. On the one hand, concrete efficiency demands low dimension; on the other, the intrinsic dimension of a gate function is large.

In the next section, we unify the schemes above into a general framework, formulating sufficient conditions for correctness and security.

4 The General Garbling Framework

We extend the techniques of RR21 to a general framework for multi-input gates, analyzing the correctness and security constraints with greater precision.

Throughout this section, we assume 3-input gates with labels sliced into two halves. Generalization to other fan-ins and numbers of slices is straightforward. The concrete instantiation is given in Section 5.

Section 4.1 introduces the garbling equation, and Section 4.2 formulates sufficient correctness and security constraints. Based on these constraints, the full garbling process is summarized in Section 4.3, and the security guarantees are stated in Section 4.4.

4.1 The Garbling Equation

We describe the evaluation of a 3-input gate $g : \{0,1\}^3 \to \{0,1\}$. The evaluator holds input colors x, y, z, input labels $A^x = (A_L^x, A_R^x)$, $B^y = (B_L^y, B_R^y)$, $C^z = (C_L^z, C_R^z)$, and garbled table $\mathbf{G}$. The evaluator must compute the output label $D^w = (D_L^w, D_R^w)$ where $w = \tilde{g}(x, y, z)$.

Inspired by the schemes reviewed in Section 3, the evaluator operates as follows. First, it makes several hash function calls, denoted collectively by $\boldsymbol{H}_{x,y,z}$, which include hashes of linear combinations of the input labels, such as $H(A^x)$, $H(B^y), H(A^x \oplus B^y)$. It then computes the output label D^w by linearly combining $\boldsymbol{H}_{x,y,z}$, the garbled table $\mathbf{G}$, and the input labels A^x, B^y, C^z:

$$D^w = \mathbf{V}^*(x,y,z)\mathbf{G} \oplus \mathbf{M}^*(x,y,z)\boldsymbol{H}_{x,y,z} \oplus \mathbf{R}(x,y,z) \begin{bmatrix} A^x \\ B^y \\ C^z \end{bmatrix},$$

where the coefficients $\mathbf{V}^*(x,y,z), \mathbf{M}^*(x,y,z), \mathbf{R}(x,y,z)$ may depend on the evaluator's color bits x, y, z.

The garbler, not knowing the evaluator's color bits in advance, must choose $(D^0, \mathbf{G})$ so that the equation holds for every color triple $(x, y, z) \in \{0,1\}^3$. Expanding $A^x = A^0 \oplus x\Delta$, and rearranging yields the *garbling equation*:

$$\underbrace{\left[\mathbf{I}\ \mathbf{V}^*(x,y,z)\right]}_{\mathbf{V}(x,y,z)} \begin{bmatrix} D^0 \\ \mathbf{G} \end{bmatrix} = \mathbf{M}(x,y,z)\boldsymbol{H} \oplus \underbrace{\left[\mathbf{R}_A\ \mathbf{R}_B\ \mathbf{R}_C\right](x,y,z)}_{\mathbf{R}(x,y,z)} \begin{bmatrix} A^0 \\ B^0 \\ C^0 \end{bmatrix} \oplus \underbrace{(x\mathbf{R}_A + y\mathbf{R}_B + z\mathbf{R}_C)(x,y,z)}_{\mathbf{R}_\Delta(x,y,z)} \Delta \oplus \tilde{g}(x,y,z)\Delta. \quad (4)$$

Here $\boldsymbol{H}$ is the union of $\boldsymbol{H}_{x,y,z}$ over all $(x, y, z) \in \{0,1\}^3$, and $\mathbf{M}(x, y, z)$ is the expansion of $\mathbf{M}^*(x, y, z)$ (obtained by inserting zeros at the appropriate positions), so that $\mathbf{M}(x, y, z)\boldsymbol{H} = \mathbf{M}^*(x, y, z)\boldsymbol{H}_{x,y,z}$.

4.2 Constraints for Correctness and Security

(Correctness Constraint) We now stack the equations over all color triples $(x, y, z) \in \{0,1\}^3$. Let $\mathbf{X}_x, \mathbf{X}_y, \mathbf{X}_z$ be the block-diagonal matrices whose blocks at (x, y, z) are $x\mathbf{I}_2, y\mathbf{I}_2, z\mathbf{I}_2$, respectively. Thus $\mathbf{X}_x\mathbf{R}_A$, for example, stacks the blocks $x\mathbf{R}_A(x, y, z)$. We write $\mathbf{T}_{\tilde{g}} = \mathcal{T}(\tilde{g}) \otimes \mathbf{I}_2$ for the stacked blocks $\tilde{g}(x, y, z)\mathbf{I}_2$.

The garbler must solve for $(D^0, \mathbf{G})$ satisfying (4) for all $(x, y, z) \in \{0,1\}^3$. Solvability over arbitrary labels, hash values, and global key Δ requires that the coefficients $\mathbf{M}, \mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C$, and $\mathbf{R}_\Delta + \mathbf{T}_{\tilde{g}}$ all lie in the column space of $\mathbf{V}$:

$$\mathbf{M}, \mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C, \underbrace{\mathbf{X}_x\mathbf{R}_A + \mathbf{X}_y\mathbf{R}_B + \mathbf{X}_z\mathbf{R}_C}_{\mathbf{R}_\Delta} + \mathbf{T}_{\tilde{g}} \in \text{colspace}(\mathbf{V}), \quad (5)$$

Here membership of a matrix in colspace($\mathbf{V}$) means that each of its columns lies in that space. In this two-slice, three-input setting, each of $\mathbf{X}_x, \mathbf{X}_y, \mathbf{X}_z$ is 16×16, while $\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C$ and $\mathbf{T}_{\tilde{g}}$ are 16×2 matrices.

Thus, given $\mathbf{V}$ and $\mathbf{M}$, correctness requires that for every permuted gate function $\tilde{g}$ there exist $\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C$ satisfying (5). Equivalently, $\mathcal{T}(\tilde{g}) \otimes \begin{bmatrix}1\\0\end{bmatrix}$ and $\mathcal{T}(\tilde{g}) \otimes \begin{bmatrix}0\\1\end{bmatrix}$ must each be expressible as $\mathbf{X}_x c_1 + \mathbf{X}_y c_2 + \mathbf{X}_z c_3 + c$ for some $c_1, c_2, c_3, c \in$ colspace($\mathbf{V}$). If every entry of $\mathbf{V}$ is affine in x, y, z, then every vector in colspace($\mathbf{V}$) is affine as well. Consequently, $\mathbf{X}_x c_1 + \mathbf{X}_y c_2 + \mathbf{X}_z c_3 + c$ has degree at most two and cannot represent a gate of degree three. This is the degree obstruction arising under the affine-coefficient assumption in the analyses of Ashur et al. [2] and Kim [13]. We therefore allow the entries of $\mathbf{V}$ to depend quadratically on the public color bits, while the garbling equation remains linear in the label slices, hash outputs, and table entries.

(Security Constraint I) $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ satisfying (5) may depend on $\tilde{g}$ and thus leak information about the permute bits. This is prevented by drawing $\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C$ uniformly at random from the solution space and revealing only the marginal values at the evaluator's color bits x, y, z. It suffices that for every $(x, y, z) \in \{0,1\}^3$, the marginal distribution

$$(\mathbf{R}_A(x, y, z), \mathbf{R}_B(x, y, z), \mathbf{R}_C(x, y, z))$$

is uniform.

(Security Constraint II) In addition to the 1-bit permute secrets α, β, γ, the λ-bit global key Δ must also remain hidden. Since Δ enters the computation of D^0 and $\mathbf{G}$ via (4), and $\mathbf{G}$ is sent to the evaluator, $\mathbf{G}$ must appear uniformly random from the evaluator's perspective.

Concretely, $\mathbf{G}$ should appear random owing to *unqueried* hash function calls. By (4), D^0 and $\mathbf{G}$ are determined as follows, assuming that $\mathbf{V}$ has independent columns, as in all constructions below:

$$\begin{bmatrix}D^0\\ \mathbf{G}\end{bmatrix} = \left(\mathbf{V}^{-1}\mathbf{M}\right) \boldsymbol{H} \oplus \text{rest}, \tag{6}$$

where $\mathbf{V}^{-1}$ is any left-inverse of $\mathbf{V}$. We require that $\mathbf{G}$ be uniform even after conditioning on $\boldsymbol{H}_{x,y,z}$ and the "rest" terms. Equivalently, the submatrix of $\mathbf{V}^{-1}\mathbf{M}$ formed by removing the first two rows and all columns indexed by $\boldsymbol{H}_{x,y,z}$ must have full row rank for every $(x, y, z) \in \{0,1\}^3$. That is, fixing $\boldsymbol{H}_{x,y,z}$ and the "rest" terms, the linear map from $\boldsymbol{H} \backslash \boldsymbol{H}_{x,y,z}$ to $\mathbf{G}$ induced by (6) is surjective.

4.3 The Full Process

Given a garbling equation satisfying both correctness and security constraints, the garbler—holding zero labels A^0, B^0, C^0, permute bits α, β, γ, global key Δ, and output wire index idx—executes the process in fig. 1.

(a) Draw $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ uniformly among solutions of

$$\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C \in \text{colspace}(\mathbf{V}), \qquad \mathbf{R}_\Delta + \mathbf{T}_{\tilde{g}} \in \text{colspace}(\mathbf{V}),$$

where $\mathbf{R}_\Delta = \mathbf{X}_x\mathbf{R}_A + \mathbf{X}_y\mathbf{R}_B + \mathbf{X}_z\mathbf{R}_C$. This guarantees solvability of the garbling equation.

(b) Solve $(D^0, \mathbf{G})$ according to eq. (4)

$$\begin{bmatrix} D^0 \\ \mathbf{G} \end{bmatrix} = \mathbf{V}^{-1} \left(\mathbf{M}\boldsymbol{H} \oplus \mathbf{R} \begin{bmatrix} A^0 \\ B^0 \\ C^0 \end{bmatrix} \oplus \mathbf{R}_\Delta \Delta \oplus \mathbf{T}_{\tilde{g}} \Delta \right),$$

where $\mathbf{R}$ and $\mathbf{R}_\Delta$ are determined by $(\mathbf{R}_A, \mathbf{R}_B, \mathbf{R}_C)$ sampled in (a), and each oracle call $H(b_0A^x \oplus b_1B^y \oplus b_2C^z)$ in $\boldsymbol{H}$ is padded to $H(b_0A^x \oplus b_1B^y \oplus b_2C^z; (\mathsf{idx}, b_0, b_1, b_2))$.

(c) One-time pad $[\mathbf{R}_A\, \mathbf{R}_B\, \mathbf{R}_C]\,(x, y, z)$ using $H(A^x, B^y, C^z; \mathsf{idx})$ for all $(x, y, z) \in \{0, 1\}^3$ and denote the ciphertext as $Z_{x,y,z}$.[a]

(d) Send to the evaluator $(\mathbf{G}, \mathbf{Z} := \{Z_{x,y,z}\}_{(x,y,z)\in\{0,1\}^3})$ as the garbled table of gate g.

Fig. 1: Steps for garbling a single gate g.

[a] Without loss of generality, we assume that $H(A, B, C)$ is truncated to fit the size of $\mathbf{R}$ before one-time pad.

Given (A^x, B^y, C^z), the evaluator decrypts $\left[\mathbf{R}_A\ \mathbf{R}_B\ \mathbf{R}_C\right](x, y, z)$ from $Z_{x,y,z}$ using $H(A^x, B^y, C^z; \mathsf{idx})$, then computes

$$\mathbf{V}^*(x, y, z)\mathbf{G} \oplus \mathbf{M}^*(x, y, z)\boldsymbol{H}_{x,y,z} \oplus \left[\mathbf{R}_A\ \mathbf{R}_B\ \mathbf{R}_C\right](x, y, z) \begin{bmatrix} A^x \\ B^y \\ C^z \end{bmatrix} \tag{7}$$

to obtain the output label $D^0 \oplus \tilde{g}(x, y, z)\Delta$.

For a full circuit, the garbler fixes one global offset Δ and processes gates in topological order, using free-XOR on XOR nodes and Figure 1 on the remaining gates. Input encoding selects each label using the semantic input and the secret permutation bit. Evaluation follows the same gate order. Each output wire carries two domain-separated decoding hashes, one for each semantic value; decoding identifies the hash matching the computed output label. Appendix A.1 specifies all four algorithms.

4.4 Correctness and Security

We use the garbling notions of Bellare, Hoang, and Rogaway [6]. The permitted leakage $\Phi(C)$ is the circuit topology and the positions of free-XOR gates. Obliviousness hides the input and gate functions beyond this leakage; privacy

additionally allows the circuit output to be revealed; authenticity prevents an evaluator from producing an accepted, incorrect output.

Theorem 1 (Security of the framework). *Suppose the garbling equations satisfy the correctness and security constraints of Section 4.2, with independent columns in* $\mathbf{V}$*. The full scheme of Construction 5, using the uncompressed controls of Figure 1, is correct, oblivious, private, and authentic in the random-oracle model. The same guarantees hold under the sliced CCR condition of Section 4.5, with its specified output-decoding hashes.*

Correctness follows by solving the equation at each gate. For security, the first security constraint makes the visible controls independent of the secret permutation, while the second lets unknown hashes mask the table. The simulator can therefore sample the table first and complete the unavailable hash values consistently. Output-decoding hashes then give privacy and authenticity. The formal definitions and hybrid proof are in Appendix A; Appendix B gives the CCR proof.

4.5 Circular Correlation Robustness for Sliced Labels

The same garbling equations work under CCR for any number of slices. Partition the global offset into s equal slices $\Delta_1, \ldots, \Delta_s$, using padding when necessary. For a label L, public tweak τ, and mask $b \in \{0,1\}^s$, the relevant oracle is

$$\mathrm{CCR}_{\Delta}^{(s)}(L;\tau;b) = H(L \oplus \Delta;\tau) \oplus \bigoplus_{a=1}^{s} b_a \Delta_a. \tag{8}$$

CCR requires this oracle to be indistinguishable from a random function when each pair (L,τ) is queried at most once, including across different masks. This lets the second security constraint absorb offset terms into unknown hash outputs. Output-decoding tags concatenate s domain-separated calls to H. Appendix B.1 gives the formal experiment and implementation details, followed by the proof of all three security properties.

4.6 A Lower Bound on Communication

We now state the communication limit imposed by these matrix conditions for arbitrary fan-in. Let $n \geq 2$ be the number of inputs, $s \geq 1$ the number of equal-width slices in a label, and r the number of transmitted ciphertext slices. Fix the public matrices $\mathbf{V}, \mathbf{M}$ to support every n-input Boolean gate, while allowing the control matrices to depend on the gate function. The stacked matrix $\mathbf{V} \in \mathbb{F}_2^{s2^n \times (s+r)}$ has independent columns, and its first s columns stack the identity matrix $\mathbf{I}_s$.

For the main table, we use the same query type as above: hashes of fixed, nonzero binary linear combinations of input labels, with public tweaks. The two hash values in each pair are available on complementary halves of the color space,

determined by the corresponding linear form. Their coefficient columns vanish wherever the respective hash value is unavailable. The separate hashes used to encrypt the control information are not part of this main-table condition.

Theorem 2 (Communication lower bound). *Any such construction satisfying the correctness and both security constraints of Section 4.2 satisfies*

$$r \geq \max\left\{ \left\lceil \frac{s(2^n - n - 1)}{n - 1} \right\rceil, \left\lceil \frac{s(2^n - 1)}{n} \right\rceil \right\}. \tag{9}$$

In particular,

$$\frac{r}{s} \geq \max\left\{ \frac{2^n - n - 1}{n - 1}, \frac{2^n - 1}{n} \right\}. \tag{10}$$

The first term already follows from correctness and security constraint II; the second additionally uses security constraint I.

The main table contributes $r\lambda/s$ bits before slice padding and control encoding. Thus, for $n = 2, 3, 4$, its communication is at least $3\lambda/2$, $7\lambda/3$, and $15\lambda/4$, respectively. Extension-field constructions obey the same bound after expansion into binary coordinates, which preserves r/s.

The proof counts linear relations among the columns of the correctness matrix. Security constraint II supplies supported hash columns, and averaging over the two equal-sized color classes defined by each linear form gives the first bound. Security constraint I additionally requires enough freedom to vary all local controls, giving the second. Appendix C gives an elementary matrix proof.

5 Constructions

Our framework reduces the construction of a secure linear garbling scheme to finding matrices that satisfy the conditions in Section 4.2. We abbreviate correctness and security constraints I and II as C, S1, and S2, respectively.

From this section onward, write $x = (x_1, \ldots, x_n)$ for the formal color variables in the garbling equations and $p = (p_1, \ldots, p_n)$ for the evaluator's actual color bits.

We pursue two approaches. For small fan-in $3 \leq n \leq 7$, we find explicit matrices by randomized search and verify them exactly. For arbitrary fan-in $n \geq 2$, we prove an existence result. Both approaches build on RR21, but use different features of its matrices. Its complementary hash pairs suggest an acyclic search space in which S2 holds automatically. Its scalar factors suggest a tensor construction whose remaining conditions can be expressed in the language of coding theory and established using known covering results.

5.1 Explicit Construction for $3 \le n \le 7$

We start with the concrete RR21 matrices in eq. (3). With the two formal input variables written as $x = (x_1, x_2)$, these matrices are

$$\mathbf{V}(x) = \begin{bmatrix} 1 & 0 & x_1 & 0 & x_1 + x_2 \\ 0 & 1 & 0 & x_2 & x_1 + x_2 \end{bmatrix},$$
$$\mathbf{M}(x) = \begin{bmatrix} 1 + x_1 & x_1 & 0 & 0 & 1 + x_1 + x_2 & x_1 + x_2 \\ 0 & 0 & 1 + x_2 & x_2 & 1 + x_1 + x_2 & x_1 + x_2 \end{bmatrix}.$$

The last three columns of $\mathbf{V}(x)$ have the form $F_j(x) = \ell_j(x)c_j$, where

$$(\ell_1, \ell_2, \ell_3) = (x_1, x_2, x_1 + x_2), \qquad (c_1, c_2, c_3) = \left(\binom{1}{0}, \binom{0}{1}, \binom{1}{1} \right).$$

Each ℓ_j also determines a pair of hash inputs. For example, the last two columns of $\mathbf{M}(x_1, x_2)$, $(1 + x_1 + x_2)\binom{1}{1}, (x_1 + x_2)\binom{1}{1}$, are the coefficients of $H(A^0 \oplus B^0)$ and $H(A^0 \oplus B^0 \oplus \Delta)$, respectively. An evaluator holding A^{p_1}, B^{p_2} can compute one of these two hash values, depending on $\ell_3(p) = p_1 + p_2$. For example, if $\ell_3(p) = 0$, only $H(A^0 \oplus B^0)$ is available.

Condition S2 (eq. (6)) requires the unavailable hash values to mask the three ciphertext slices independently. Fix the evaluator's color p. In pair j, the unavailable hash has coefficient

$$(\ell_j(x) + \ell_j(p))c_j.$$

In particular, this coefficient vanishes at $x = p$. If $p_1 = 0$, for example, $H(A^0 \oplus \Delta)$ is unavailable and its coefficient function is $x_1 c_1$. Although this coefficient is zero at the evaluator's input, its column in the stacked system can still randomize the ciphertext slices.

The unavailable-hash submatrix is

$$\begin{aligned} \mathbf{M}_p &= \begin{bmatrix} \ell_1(x)c_1 & \ell_2(x)c_2 & \ell_3(x)c_3 \end{bmatrix} + \begin{bmatrix} \ell_1(p)c_1 & \ell_2(p)c_2 & \ell_3(p)c_3 \end{bmatrix} \\ &= \begin{bmatrix} x_1 & 0 & x_1 + x_2 \\ 0 & x_2 & x_1 + x_2 \end{bmatrix} + \begin{bmatrix} p_1 & 0 & p_1 + p_2 \\ 0 & p_2 & p_1 + p_2 \end{bmatrix}, \end{aligned}$$

and hence

$$\mathbf{V}^{-1}\mathbf{M}_p = \left[\begin{array}{ccc} p_1 & 0 & p_1 + p_2 \\ 0 & p_2 & p_1 + p_2 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right],$$

where the equality follows because the columns $\ell_j(x)c_j$ are the rightmost columns of $\mathbf{V}(x)$, while the constant columns $\ell_j(p)c_j$ lie in the span of its first two columns. The lower block of $\mathbf{V}^{-1}\mathbf{M}_p$ has full row rank for every color vector $p \in \{0,1\}^2$, which guarantees S2 in RR21's scheme.

The difficulty with extending this example lies in *correctness*. If every column remains linear in the input bits, multiplying a column by one input bit produces

a polynomial of degree at most two. Such columns suffice for every two-input function, but cannot represent the three-input function $x_1x_2x_3$. To obtain higher-degree terms, we allow a column to depend on earlier columns before multiplying by its linear form. We will see that this replaces the identity matrix used to establish S2 with an upper triangular matrix whose diagonal entries are still one.

A recursive matrix construction. We now extend this construction to n Boolean inputs and s slices, writing $x = (x_1, \dots, x_n) \in \{0,1\}^n$ for the formal color vector. Let $N = 2^n$, and let $K = \mathbb{F}_q$ be a finite field of characteristic two. All matrices in this subsection are over K, while the input coordinates remain Boolean. We aim for r ciphertext slices, so the matrix $\mathbf{V}(x)$ has $d = s+r$ columns. For $q = 2^t$, replacing every field coordinate by its t binary coordinates multiplies both s and r by t, leaving the communication ratio r/s unchanged.

For each $1 \le j \le r$, choose a nonzero binary linear form $\ell_j(x) = \sum_{i=1}^{n} \ell_{j,i} x_i$, a constant vector $c_j \in K^s$, and coefficients $a_{ij} \in K$ for $i < j$. Define

$$\begin{aligned} G_j(x) &= c_j + \sum_{i<j} a_{ij} F_i(x), \qquad F_j(x) = \ell_j(x) G_j(x), \\ \mathbf{A}(x) &= [F_1(x) \ \cdots \ F_r(x)], \qquad \mathbf{V}(x) = [\mathbf{I}_s \ \mathbf{A}(x)]. \end{aligned} \tag{11}$$

Stacking $\mathbf{V}(x)$ over all N inputs gives the $Ns \times (s+r)$ matrix $\mathbf{V}$. We require its columns to be independent. The ordering $i < j$ ensures that these dependencies are acyclic.

For each node, use the complementary coefficients

$$h_{j,1}(x) = \ell_j(x) G_j(x) = F_j(x), \qquad h_{j,0}(x) = (1 + \ell_j(x)) G_j(x)$$

for hash calls on linear combinations of input labels, as in RR21. Stacking their evaluations gives the two columns of $\mathbf{M}$ for this hash pair. We allow repeated linear forms $\ell_j = \ell_i, i < j$ and include the node index in the hash domain so that distinct nodes use distinct hash pairs. The first coefficient vanishes whenever $\ell_j(x) = 0$, and the second whenever $\ell_j(x) = 1$. Both belong to colspace($\mathbf{V}$), because

$$h_{j,0}(x) = c_j + F_j(x) + \sum_{i<j} a_{ij} F_i(x).$$

The argument establishing S2 in RR21 extends to this acyclic construction. The bottom r entries of $\mathbf{V}^{-1} h_{j,1}, \mathbf{V}^{-1} h_{j,0}$ are e_j and $e_j + \sum_{i<j} a_{ij} e_i$, respectively, where e_j is the j-th unit vector in K^r. Let $\mathbf{D}$ be the $r \times r$ matrix with $\mathbf{D}_{ij} = a_{ij}$ for $i < j$ and zero elsewhere. If the evaluator has color vector p, the unqueried hash values mask the table entries through the coefficient matrix

$$\mathbf{I}_r + \mathbf{D} \operatorname{diag}(\ell_1(p), \dots, \ell_r(p)) = \begin{pmatrix} 1 & * & * & \cdots \\ 0 & 1 & * & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{12}$$

This matrix is invertible for every p, proving S2. Its entries above the diagonal may depend on the input, but its diagonal entries are always one. This argument works for every choice of constants and edge coefficients, provided the proposed columns of $\mathbf{V}$ are independent. It remains to check correctness and S1.

Dependencies provide the missing high-degree terms. For example, $F_1 = x_1c_1$ and $F_2 = x_2(c_2 + F_1)$ give $F_2 = x_2c_2 + x_1x_2c_1$. A path through three nodes can similarly create a product of three linear forms. Cancellation may remove such terms, so long paths alone do not guarantee correctness. They do, however, explain why a DAG is a useful search space: S2 holds by construction, while dependencies let the columns acquire the degrees needed for C.

Randomized search. For fixed (n, K, s, r), we randomly sample the nonzero binary forms, constant vectors, and permitted edge coefficients in (11). We reject dependent columns and complete candidates that fail C or S1 at any input. In incremental searches, necessary rank bounds also eliminate prefixes whose rank deficits cannot be repaired by the remaining columns. Rejected proposals are redrawn, with restarts when the proposal budget is exhausted. S2 follows from the triangular construction, and every accepted candidate is independently verified for all three conditions at every input. Appendix D gives the exact rank tests, sampler configurations, and measured search and verification costs.

Verified constructions. Randomized search gives the following finite constructions, using the slice parameters s, r of Theorem 2.

Theorem 3 (Verified finite constructions). *For every $3 \le n \le 7$, there are fixed binary matrices $\mathbf{V}, \mathbf{M}$ satisfying C, S1, and S2 for every n-input Boolean gate, with s label slices and r ciphertext slices given by*

$$(s, r) = \begin{cases} (3, 7), & n = 3, \\ (8, 30), & n = 4, \\ (n-1, 2^n - n - 1), & 5 \le n \le 7. \end{cases}$$

These constructions attain

$$\frac{r}{s} = \max\left\{ \frac{2^n - n - 1}{n - 1}, \frac{2^n - 1}{n} \right\}.$$

Thus their main-table communication $r\lambda/s$ is optimal under the conditions of Theorem 2, before control encoding and slice padding.

The literal coefficients and their exact verification establish this theorem; see Appendix E.

Table 1 lists these parameters and includes RR21 at $n = 2$ for reference.

The three- and four-input matrices below are the fixed instances used in the control-encoding analysis. The searches independently recover their parameters, but the optimized control encodings are specific to the chosen matrices.

Table 1: Verified finite constructions. The dimensions in the middle column are binary; native $\mathbb{F}_4$ dimensions for $n = 4$ are $(s, r) = (4, 15)$.

n	coefficient field	binary (s, r)	communication term
2	$\mathbb{F}_2$	$(2, 3)$	$3\lambda/2$
3	$\mathbb{F}_2$	$(3, 7)$	$7\lambda/3$
4	$\mathbb{F}_4$	$(8, 30)$	$15\lambda/4$
5	$\mathbb{F}_2$	$(4, 26)$	$13\lambda/2$
6	$\mathbb{F}_2$	$(5, 57)$	$57\lambda/5$
7	$\mathbb{F}_2$	$(6, 120)$	20λ

The three-input matrices. The fixed three-input instance has $s = 3$, $r = 7$, and $K = \mathbb{F}_2$. Its forms and constant columns are

$$(\ell_1, \dots, \ell_7) = (x_1, x_2, x_3, x_1 + x_2, x_2 + x_3, x_1 + x_2 + x_3, x_1 + x_3),$$
$$\mathbf{C} = [c_1 \ \cdots \ c_7] = \begin{bmatrix} 0\,1\,1\,1\,0\,1\,0 \\ 0\,0\,1\,1\,1\,0\,1 \\ 1\,0\,0\,1\,1\,1\,0 \end{bmatrix}. \tag{13}$$

Only $\mathbf{D}_{1,2} = \mathbf{D}_{2,4} = \mathbf{D}_{3,4} = 1$ are nonzero. In particular, $F_2 = x_2(c_2 + F_1)$ and $F_4 = (x_1 + x_2)(c_4 + F_2 + F_3)$. To display the matrices compactly, write $\ell_j = \ell_j(x)$ and $\bar{\ell}_j = 1 + \ell_j(x)$. Expanding the recursion and simplifying with $x_i^2 = x_i$ gives

$$\mathbf{V}(x) = \left[\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 0 & \ell_2 & \ell_3 & \ell_4\bar{\ell}_5 & 0 & \ell_6 & 0 \\ 0 & 1 & 0 & 0 & 0 & \ell_3 & \ell_4\bar{\ell}_3 & \ell_5 & 0 & \ell_7 \\ 0 & 0 & 1 & \ell_1 & \ell_1\ell_2 & 0 & \ell_4 & \ell_5 & \ell_6 & 0 \end{array}\right]. \tag{14}$$

The columns of $\mathbf{M}(x)$ are ordered as $h_{1,0}, h_{1,1}, \dots, h_{7,0}, h_{7,1}$, with each adjacent pair corresponding to the two hash calls associated with ℓ_j:

$$\mathbf{M}(x) = \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc} 0 & 0 & \bar{\ell}_2 & \ell_2 & \bar{\ell}_3 & \ell_3 & \bar{\ell}_4\bar{\ell}_5 & \ell_4\bar{\ell}_5 & 0 & 0 & \bar{\ell}_6 & \ell_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bar{\ell}_3 & \ell_3 & \bar{\ell}_4\bar{\ell}_3 & \ell_4\bar{\ell}_3 & \bar{\ell}_5 & \ell_5 & 0 & 0 & \bar{\ell}_7 & \ell_7 \\ \bar{\ell}_1 & \ell_1 & \ell_1\bar{\ell}_2 & \ell_1\ell_2 & 0 & 0 & \bar{\ell}_4\bar{\ell}_2 & \ell_4 & \bar{\ell}_5 & \ell_5 & \bar{\ell}_6 & \ell_6 & 0 & 0 \end{array}\right]. \tag{15}$$

At evaluation, substitute $x = p$; in pair j, only the hash indexed by $\ell_j(p)$ is available, and the other coefficient vanishes. Stacking these matrices over the eight inputs gives $\mathbf{V} \in \mathbb{F}_2^{24\times 10}$ and $\mathbf{M} \in \mathbb{F}_2^{24\times 14}$. Exact verification confirms C and S1 at all eight inputs; S2 follows from the upper triangular matrix in (12).

The four-input matrices. For the four-input instance, let $K = \mathbb{F}_2[\omega]/(\omega^2+\omega+1)$, with native parameters $s = 4, r = 15$. Table 2 lists the binary form masks and the coefficients of $\mathbf{C}, \mathbf{D}$. A mask $\sum_i m_i 2^{i-1}$ represents $\sum_i m_i x_i$, and $\mathbf{C}_{aj}$ is coordinate a of c_j. These data determine the full 64×19 evaluation matrix through (11).

Table 2: The fixed four-input coefficients over $\mathbb{F}_4$. Dashes mark nonexistent indices; all unlisted entries of $\mathbf{D}$ are zero.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
mask	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9
$\mathbf{C}_{1j}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\mathbf{C}_{2j}$	0	ω^2	1	ω^2	0	0	1	ω^2	0	ω	1	ω	0	ω^2	0
$\mathbf{C}_{3j}$	1	ω^2	ω^2	0	1	ω^2	ω^2	1	ω^2	0	1	ω	1	ω	1
$\mathbf{C}_{4j}$	ω^2	0	0	0	ω^2	ω^2	0	ω	ω	0	1	0	0	ω	ω
$\mathbf{D}_{j-1,j}$	–	ω	1	1	1	1	ω^2	ω^2	1	ω^2	1	1	1	ω	ω^2
$\mathbf{D}_{j-2,j}$	–	–	ω^2	ω^2	1	1	ω	ω^2	ω^2	ω^2	ω^2	1	1	ω	ω^2

Exact verification confirms C and S1 at all sixteen inputs; the same triangular argument establishes S2. Expanding $u + v\omega$ into (u, v) gives the exact binary construction with $(s, r) = (8, 30)$. This is a field-linear construction expressed in binary coordinates, rather than a binary construction with four slices and fifteen ciphertexts.

5.2 General Construction for Any $n \geq 2$

For arbitrary fan-in, we use a tensor form: each selected scalar function $m(x)$ contributes the block $m(x)\mathbf{I}_s$, repeating the same coefficients in every output coordinate. Choosing Boolean monomials reduces correctness and security to a combinatorial condition on the selected sets of variables.

Selecting monomials. Work over $\mathbb{F}_2$, and let $[n] = \{1, \ldots, n\}$. For each subset $I \subseteq [n]$, define the Boolean monomial

$$m_I(x) = \prod_{i \in I} x_i, \qquad m_\varnothing(x) = 1.$$

The evaluations of all 2^n monomials form a basis of the scalar truth tables on $\{0,1\}^n$. In coding terminology, their span is the full Reed–Muller code $\mathrm{RM}(n, n)$; $\mathrm{RM}(t, n)$ denotes the evaluations of multilinear polynomials of degree at most t.

Choose a family $\mathcal{F} \subseteq 2^{[n]}$ containing $\varnothing$, write $b = |\mathcal{F}|$, and order its members with $\varnothing$ first. We use

$$\mathbf{V}(x) = \big[\, m_I(x)\mathbf{I}_s \,\big]_{I \in \mathcal{F}}. \tag{16}$$

Thus $\mathbf{V}(x)$ has s rows and sb columns, of which the first s represent the output label. Stacking these matrices over all inputs gives $\mathbf{V}$ as in the framework. Its columns are independent, so the number of transmitted slices is $r = s(b-1)$. Equivalently, if $B_\mathcal{F} = \mathrm{span}_{\mathbb{F}_2}\{m_I : I \in \mathcal{F}\}$, the column space consists of all tables $\sum_{I \in \mathcal{F}} m_I(x) v_I$ with $v_I \in \mathbb{F}_2^s$. This is the tensor product $B_\mathcal{F} \otimes \mathbb{F}_2^s$: every output coordinate independently uses the same scalar code. The code $B_\mathcal{F}$ need not be a degree-truncated Reed–Muller code.

The required condition on $\mathcal{F}$ is particularly concrete. It must contain $\varnothing$ and all singletons, and satisfy

$$\text{for every nonempty } J \subseteq [n], \quad J \setminus \{i\} \in \mathcal{F} \text{ for some } i \in J. \tag{17}$$

We call $J \setminus \{i\}$ a *parent* of J, since its monomial becomes m_J after multiplication by x_i. The condition applies to selected sets as well as unselected ones; it does not require every subset of a selected set to be selected.

Proposition 1. *Suppose $\mathcal{F}$ contains $\varnothing$ and all singletons and satisfies* (17). *Then* (16) *admits hash coefficients satisfying C, S1, and S2, for every $s \geq 1$, with $r = s(|\mathcal{F}| - 1)$.*

The parent condition supplies every nonconstant monomial needed for C. Including the constant and singletons lets us change any local control value and compensate elsewhere without affecting correctness, which gives S1. For S2, pair each selected nonconstant monomial with a selected parent: ordering parents before children again gives triangular hash masking with a unit diagonal. The full proof is in Appendix F.

A three-input example. At $n = 3$, choose

$$B_{\mathcal{F}} = \operatorname{span}\{1, x_1, x_2, x_3, x_1x_2\},$$
$$\mathbf{V}(x) = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_1x_2 \end{bmatrix} \otimes \mathbf{I}_s.$$

The linear monomials supply all quadratics, and the selected x_1x_2 supplies the cubic. For $s = 1$, use four complementary hash pairs with input indices $1, 2, 3, 2$ and distinct slot tweaks. The selected monomials have parents $1, 1, 1, x_1$, respectively. In the ordered table basis (x_1, x_2, x_3, x_1x_2), the unknown hashes at color p act through

$$\begin{bmatrix} 1 & 0 & 0 & p_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix has rank four at every color; tensoring with $\mathbf{I}_s$ gives rank $4s$. Appendix F.2 lists the hash coefficients and derives the matrix. The example illustrates the general mechanism; its cost 4λ before controls exceeds that of the dedicated three-input construction.

Existence of small families. One selected monomial can supply several targets by multiplication with different input coordinates. We therefore seek a family with only $O(2^n/n)$ selected monomials. This resembles a covering-code problem, but the direction of the covering matters: a parent must be obtained by deleting one element from the target.

An upward radius-one *asymmetric cover* $\mathcal{C}_t \subseteq 2^{[t]}$ requires that every set either belongs to $\mathcal{C}_t$ or has a parent in $\mathcal{C}_t$. Cooper, Ellis, and Kahng [8] prove that such covers have size $O(2^t/t)$; their downward orientation is equivalent by

complementing all sets. A cover alone does not guarantee (17), because a selected set can cover itself without having a selected parent.

A product construction followed by adding the missing parents and singletons repairs this gap while retaining size $O(2^n/n)$. Applying the cited bound gives the following theorem; the repair and constant calculation appear in Appendix F.

Theorem 4 (General construction). *For every $n \geq 2$, there exists a family $\mathcal{F}$ satisfying the conditions of Proposition 1 with*

$$|\mathcal{F}| - 1 \leq 22 \cdot 2^n/n.$$

Consequently, the framework admits a free-XOR-compatible construction for every n-input, single-output gate whose communication is at most

$$22 \cdot 2^n \lambda/n + 2^n$$

bits in the random-oracle model, with nonadaptive queries within each gate.

For this bound, take $s = 1$, so no slice padding is needed. Writing $b = |\mathcal{F}|$, the syndrome encoder of Section 6.1 gives the precise per-gate cost

$$(b-1)\lambda + 2^n - b. \tag{18}$$

The two terms account for the main table and the correction of the $2^n - b$ unselected monomial coefficients, respectively. Appendix F.5 gives the encoder, cost derivation, and joint security proof.

The theorem establishes the asymptotic scope of the framework. Given a suitable cover, all matrices and their security witnesses are explicit; the covering existence result does not provide a general running-time bound for finding the cover or a concrete advantage over the finite constructions.

6 Concrete Efficiency

Concrete communication depends on the control overhead as well as the main garbled table, and on which subcircuits can usefully be replaced. We address both questions for the three-input construction.

6.1 Reducing the Constant Overhead

The control matrices $\mathbf{R}_i(p)$ are linked by the garbling equation, so their full tables contain redundant information. There are n matrices of size $s \times s$ at each of 2^n colors: sending every entry would cost $ns^2 2^n$ bits. For our three-input construction, this is $3 \cdot 9 \cdot 8 = 216$ bits. Instead, we generate random controls from input-label hashes and send only a correction that makes them satisfy the garbling equation. The method applies to any fan-in; we use the three-input instance in (13) as a running example.

The correctness constraints. Put $N = 2^n$ and $d = s + r$, so the stacked matrix $\mathbf{V}$ has Ns rows and d independent columns. Choose a public matrix

$$\mathbf{P} \in \mathbb{F}_2^{u \times Ns}, \qquad u = Ns - d, \qquad \ker \mathbf{P} = \mathrm{colspace}(\mathbf{V}).$$

Thus $\mathbf{P}\mathbf{y} = 0$ exactly when $\mathbf{y}$ can already be expressed using $\mathbf{V}$. Multiplication by $\mathbf{P}$ records only the missing coordinates, which we call a *syndrome*.

Write $\mathbf{R}_i = \mathbf{V}\mathbf{Q}_i$, with $\mathbf{Q}_i \in \mathbb{F}_2^{d \times s}$, and collect the control coefficients in $F = (\mathbf{Q}_1, \ldots, \mathbf{Q}_n)$. Let $\mathbf{X}_i$ be block diagonal with blocks $x_i\mathbf{I}_s$, and let $\mathbf{T}_{\tilde{g}}$ stack the blocks $\tilde{g}(x)\mathbf{I}_s$. Correctness (5) now becomes

$$\underbrace{\sum_{i=1}^{n} \mathbf{P}\mathbf{X}_i\mathbf{V}\mathbf{Q}_i}_{\mathcal{N}(F)} = \underbrace{\mathbf{P}\mathbf{T}_{\tilde{g}}}_{\gamma(\tilde{g})}. \tag{19}$$

Here $\mathcal{N}$ maps the coefficient space $\mathcal{A} = (\mathbb{F}_2^{d \times s})^n$ onto the syndrome space $\mathcal{Q} = \mathbb{F}_2^{u \times s}$, by C. In our three-input example, $s = 3$, $d = 10$, and $u = 24 - 10 = 14$, so the right-hand side has $14 \cdot 3 = 42$ bits.

Generating the controls and sending a correction. Use fresh, domain-separated input-label hash bits to form a random coefficient tuple $h = (h_1, \ldots, h_n) \in \mathcal{A}$. As with the main-table hashes, each bit multiplies a control table that vanishes wherever its hash input is unavailable. This lets the evaluator compute $\mathbf{V}(p)h_i$ from its active labels. A hash bit may contribute to several control matrices, allowing their masks to be chosen jointly.

Choose a public linear subspace $\mathcal{D} \subseteq \mathcal{Q}$ containing every gate syndrome $\gamma(\tilde{g})$, and choose the masks so that $\mathcal{N}(h)$ also lies in $\mathcal{D}$. Fix a public linear map $J : \mathcal{D} \to \mathcal{A}$ with $\mathcal{N}J = I_{\mathcal{D}}$. The garbler sends

$$\mathbf{z} = \gamma(\tilde{g}) + \mathcal{N}(h), \qquad F = h + J\mathbf{z}. \tag{20}$$

The correction cancels the error in the initial controls, since $\mathcal{N}(F) = \gamma(\tilde{g})$. We transmit only the coordinates of $\mathbf{z}$ in a fixed basis of $\mathcal{D}$, costing $\dim \mathcal{D}$ bits. The basis and J depend on the construction, not on the gate being garbled.

The full space $\mathcal{D} = \mathcal{Q}$ is always an option under C, S1, and S2. To use a smaller space, we check two further facts: at every color, the unknown hash bits mask every transmitted coordinate; and the corrected controls retain the original uniform distribution over correct solutions. These are linear rank checks, stated precisely in Appendix G.1.

For our three-input matrices, the gate syndromes span only seven dimensions of the 42-bit space, because all three columns come from the same Boolean truth table. This suggests choosing the masks jointly, but seven coordinates do not suffice under their required supports. The finite calculation gives a 35-dimensional message space satisfying both checks at all eight colors. Thus the same encoder sends 35 bits. Appendix G.3 gives the calculation and proves minimality within this fixed linear encoding format.

Recovering the local controls. Let $J_i(\mathbf{z})$ be the i-th component of $J\mathbf{z}$. The evaluator uses the following small garbling equation:

$$\mathbf{R}_i(p) = \underbrace{\mathbf{V}(p)h_i}_{\text{known hash mask}} + \underbrace{\mathbf{V}(p)J_i(\mathbf{z})}_{\text{public correction}} . \tag{21}$$

The unknown hashes have zero coefficients at p, so the first term can be computed without reconstructing h_i. The second term is determined by the transmitted coordinates of $\mathbf{z}$. For example, at $p = 0$ in our three-input construction, $\mathbf{V}(0) = [\mathbf{I}_3\ \mathbf{0}_{3\times 7}]$: the equation uses only the first three rows of h_i and $J_i(\mathbf{z})$ to recover the 3×3 matrix $\mathbf{R}_i(0)$. The same 35-bit message supplies the correction for all three controls, at every color.

With equal slice widths and padding, the resulting component count is

$$C_3(\lambda) = 7\lceil \lambda/3 \rceil + 35, \tag{22}$$

namely 336 and 637 bits at $\lambda = 128$ and 256. This encoding covers the full gate table. If truth-table parity may be public, the count becomes 29 bits, or 30 including that parity bit; our evaluation uses 35. The same method gives 355 control bits for the binary expansion of our four-input construction, compared with 360 bits for the simpler native $\mathbb{F}_4$ encoding.

Security. The masking argument follows Appendix A; Appendix G.1 gives the common independent-pad argument, and Appendix F.5 gives the joint RO proof for the unsliced full-syndrome encoder. The 35-bit instantiation is used for component communication estimates; the CCR analysis retains uncompressed controls.

6.2 Circuit Evaluation Results

A smaller three-input garbling component can reduce circuit communication only if useful groups of gates can be replaced without losing intermediate values needed elsewhere. We investigate this question by comparing mixed two- and three-input circuits with the best two-input candidates collected for the same functions. We use existing circuit representations and simple local replacements, aiming to identify opportunities for savings with a modest amount of synthesis effort.

Cost model and binary references. We count nonlinear gate components and treat XOR, NOT, and wiring as free. Our two-input cost is the one stated in the RR21 abstract [20]:

$$C_2(\lambda) = 1.5\lambda + 5, \qquad C_3(\lambda) = 7\lceil \lambda/3 \rceil + 35.$$

For $\lambda = 128, 256, 512$, the pair (C_2, C_3) is $(197, 336)$, $(389, 637)$, and $(773, 1232)$, respectively. Replacing two AND gates by one three-input gate saves 58, 141, or 314 bits when the intermediate wire has no other consumer. We use the

same C_2 for every binary nonlinear node on both sides of the comparison. The counts include control information and slice padding, and exclude input OT, input and output label delivery, protocol framing, and computation time. They are component communication estimates for public circuit descriptions.

For each function, the binary reference is the circuit with the fewest AND gates in our collected pool. This pool includes the original benchmark netlists, binary realizations of EPFL LUT6 size/depth results [16], arithmetic netlists from a published synthesis artifact [23], specialized cryptographic constructions, and our verified binary simplifications. Any binary expansion of a mixed candidate also enters the reference pool. Thus an improvement in the underlying binary circuit is available to both sides of the comparison. "Best" refers to this pool; we do not establish globally minimum AND counts.

A direct approach to mixed circuits. Our mixed candidates combine verified networks reconstructed from an existing three-input synthesis flow [23, 24] with candidates produced by local replacements. We instantiate all of them with the same C_2 and C_3 costs. The experiment therefore measures the opportunities offered by existing representations and small modifications to them.

For local replacements, we enumerate single-output regions with at most three boundary wires, including regions containing XOR gates. Each replacement is checked exhaustively, must lower communication, and must preserve all externally used intermediate wires. We try several greedy orders and also reassociate unshared AND regions. Appendix H gives the search limits and implementation details. This procedure finds local opportunities; it does not jointly optimize the circuit representation and the placement of three-input gates.

Let B_f denote the selected binary AND count for benchmark f, and write (A, T) for a mixed candidate's binary and ternary nonlinear gate counts. We select the least cost among $B_f C_2$ and all verified mixed candidates at AC_2+TC_3. Retaining the binary option records zero improvement when no mixed candidate is cheaper. Each network is repriced at all three security parameters, irrespective of the parameter used when it was synthesized; the $\lambda = 512$ results do not use additional synthesis runs.

Verification and scope. Accepted mixed candidates have a complete all-output equivalence check or a complete chain of exhaustively verified local replacements from an equivalent source. The local checks account for all external wire uses. An independent checker reinterprets the final netlists and recomputes costs. The artifact retains the source circuits, parameters, netlists, and verification outcomes.

The pool contains 1,027 candidate records for 50 benchmarks. For 42 benchmarks, including every control and arithmetic instance, the selected binary and mixed candidates have complete equivalence evidence. The remaining references comprise two AES circuits using 29-AND S-boxes [18] with structural and finite-test evidence, five stronger published MPC counts, and a SHA-256 count conditional on its interface adaptation. We retain these stronger references in the denominator. When such a reference is selected as a fallback, it records no established improvement, rather than a newly verified implementation.

Results. For a category $\mathcal{F}$, we report

$$1 - \frac{\sum_{f\in\mathcal{F}} C^*_{\text{ours}}(f)}{\sum_{f\in\mathcal{F}} B_f C_2},$$

where $C^*_{\text{ours}}(f)$ is the selected cost, including binary fallback. Each listed circuit is counted once. This compares total communication, rather than averaging per-instance percentages.

We also measure the proportion of binary AND gates represented by three-input gates. Each ternary gate in the selected networks has minimum two-input AND cost two. Thus a gate-by-gate binary expansion of a selected circuit with counts (A_f, T_f) has $A_f + 2T_f$ AND gates, of which $2T_f$ can be grouped into ternary gates. We report

$$\eta_f = \frac{2T_f}{A_f + 2T_f}, \qquad \eta_{\mathcal{F}} = \frac{2\sum_{f\in\mathcal{F}} T_f}{\sum_{f\in\mathcal{F}}(A_f + 2T_f)}. \tag{23}$$

Binary fallbacks contribute $T_f = 0$. This proportion describes the selected circuit's binary expansion, which need not be the best collected binary reference with count B_f. It measures the AND gates covered by the replacements, rather than the reduction in gate count or communication, and is not an upper bound on what further synthesis could achieve.

Table 3: Our selected component cost relative to the strongest collected binary reference, including binary fallback. Positive counts are for $\lambda = 128/256/512$. The η column reports AND coverage at $\lambda = 128$; the last three columns report communication savings.

Category	Instances	Positive	η	$\lambda = 128$	$\lambda = 256$	$\lambda = 512$
Control	10	9/9/9	40.31%	5.924%	7.296%	8.178%
Arithmetic	9	7/7/7	9.77%	1.429%	1.762%	1.975%
Crypto/comparators	11	2/2/2	2.64%	0.031%	0.121%	0.179%
MPC	20	10/11/11	5.64%	0.830%	1.022%	1.146%
All	50	28/29/29	7.03%	0.903%	1.143%	1.297%
Complete-evidence subset	42	28/28/28	12.59%	1.619%	2.049%	2.325%

Control circuits have the largest coverage and communication gains (Table 3): 3,692 of 9,158 AND gates are grouped, or 40.31%. Overall coverage rises from 7.03% at $\lambda = 128$ to 7.04% at $\lambda = 256, 512$, when one more MPC benchmark selects a mixed circuit. MPC coverage then becomes 5.66%; the other categories are unchanged.

For control circuits, the aggregate savings are 5.92%, 7.30%, and 8.18% at $\lambda = 128, 256, 512$, respectively. Nine of the ten instances improve at each parameter; the largest individual saving is 16.41% on `arbiter` at $\lambda = 512$. Table 4 gives the individual counts and savings.

Table 4: Control-circuit savings (%) over the best binary references with AND counts B. Selected counts (A, T) and AND coverage η (%) are identical across all three parameters. All rows allow binary fallback and have complete equivalence evidence.

Circuit	B	A	T	η	$\lambda = 128$	$\lambda = 256$	$\lambda = 512$
`arbiter`	766	143	312	81.36	11.86	14.63	16.41
`cavlc`	394	154	120	60.91	8.97	11.04	12.37
`ctrl`	45	31	7	31.11	4.58	5.64	6.32
`dec`	292	292	0	0.00	0.00	0.00	0.00
`i2c`	557	379	89	31.96	4.70	5.79	6.49
`int2float`	85	51	17	40.00	5.89	7.25	8.12
`mem_ctrl`	4695	2757	969	41.28	6.08	7.48	8.38
`priority`	323	129	97	60.06	8.84	10.89	12.20
`router`	93	23	35	75.27	11.08	13.64	15.29
`voter`	1907	1507	200	20.98	3.09	3.80	4.26
Total	9157	5466	1846	**40.31**	**5.92**	**7.30**	**8.18**

The largest coverage is 81.36% on `arbiter`: its 312 ternary gates replace 624 AND gates in a 767-AND expansion. Its best collected binary reference has 766 AND gates, illustrating why coverage and savings use different denominators.

On `voter`, 200 ternary gates replace 400 of the 1,907 reference AND gates. Its savings are therefore $200(2C_2 - C_3)$: 11,600, 28,200, and 62,800 bits at $\lambda = 128, 256, 512$.

Outside control circuits, gains are modest; aggregate savings over all 50 benchmarks are 0.90%, 1.14%, and 1.30%. A favorable gate cost alone therefore does not ensure substantial circuit-level savings.

These experiments reuse available netlists and bounded local rewrites. A synthesis flow designed around the mixed gate costs could consider larger regions, alternative decompositions, or coordinated replacements. Such a study is beyond our scope: the current results show possible savings, without establishing the best mixed circuits or the full extent of the benefit.

References

1. Ashur, T., Hazay, C., Satish, R.: Garbling 3-input AND gates. In: CFAIL 2023. Santa Barbara, CA, USA (Aug 19, 2023)
2. Ashur, T., Hazay, C., Satish, R.: On the feasibility of sliced garbling. Cryptology ePrint Archive, Report 2024/389 (2024), https://eprint.iacr.org/2024/389
3. Baek, C., Kim, T.: Can we beat three halves lower bound? (Im)possibility of reducing communication cost for garbled circuits. Designs, Codes and Cryptography **93**(6), 2073–2105 (2025). https://doi.org/10.1007/s10623-025-01577-0, https://doi.org/10.1007/s10623-025-01577-0
4. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for Boolean and arithmetic circuits. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S.

(eds.) ACM CCS 2016. pp. 565–577. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978410
5. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press (May 1990). https://doi.org/10.1145/100216.100287
6. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382279
7. Boyar, J., Peralta, R., Pochuev, D.: On the multiplicative complexity of boolean functions over the basis (cap, +, 1). Theor. Comput. Sci. **235**(1), 43–57 (2000). https://doi.org/10.1016/S0304-3975(99)00182-6, https://doi.org/10.1016/S0304-3975(99)00182-6
8. Cooper, J.N., Ellis, R.B., Kahng, A.B.: Asymmetric binary covering codes. Journal of Combinatorial Theory, Series A **100**(2), 232–249 (2002), https://arxiv.org/abs/math/0309081
9. Fan, L., Lu, Z., Zhou, H.S.: Column-wise garbling, and how to go beyond the linear model. Cryptology ePrint Archive, Report 2024/415 (2024), https://eprint.iacr.org/2024/415
10. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast garbling of circuits under standard assumptions. Journal of Cryptology **31**(3), 798–844 (Jul 2018). https://doi.org/10.1007/s00145-017-9271-y
11. Heath, D., Kolesnikov, V., Ng, L.K.L.: Garbled circuit lookup tables with logarithmic number of ciphertexts. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part V. LNCS, vol. 14655, pp. 185–215. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58740-5_7
12. Januzelli, J., Rosulek, M., Roy, L.: Lower bounds for garbled circuits from Shannon-type information inequalities. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025, Part IV. Lecture Notes in Computer Science, vol. 16003, pp. 589–618. Springer (2025). https://doi.org/10.1007/978-3-032-01884-7_19
13. Kim, T.: Analysis on sliced garbling via algebraic approach. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part VIII. LNCS, vol. 15491, pp. 237–265. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0944-4_8
14. Kolesnikov, V., Mohassel, P., Rosulek, M.: FleXOR: Flexible garbling for XOR gates that beats free-XOR. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 440–457. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_25
15. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Berlin, Heidelberg (Jul 2008). https://doi.org/10.1007/978-3-540-70583-3_40
16. LSILS: EPFL combinational benchmark suite: Best results. Repository, revision 82d8cc6 (2026), lUT6 size and depth results
17. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: EC. pp. 129–139. ACM (1999)
18. NIST Circuit Complexity Team: AES forward S-box: 29 AND gates. Official circuit file (2026), retrieved 14 September 2026
19. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912,

pp. 250–267. Springer, Berlin, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_15
20. Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 94–124. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_5
21. Xu, F., Hu, H., Xu, C.: Bitwise garbling schemes: A model with $3\lambda/2$-bit lower bound of ciphertexts. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025, Part IV. Lecture Notes in Computer Science, vol. 16003, pp. 459–488. Springer (2025). https://doi.org/10.1007/978-3-032-01884-7_15, https://doi.org/10.1007/978-3-032-01884-7_15
22. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982). https://doi.org/10.1109/SFCS.1982.38
23. Yu, M.: Mockturtle fork: HOST synthesis code, cache, and benchmarks. Repository, revision fe8c6f1
24. Yu, M., De Micheli, G.: Generating lower-cost garbled circuits: Logic synthesis can help. In: 2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 304–314 (2023), https://si2.epfl.ch/demichel/publications/archive/2023/HOST2023.pdf
25. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_8

A Circuit-Level Algorithms and Security in the RO Model

This appendix gives the full algorithms and the random-oracle proof of Theorem 1. We retain the three-input, two-slice notation of Section 4; the argument uses only its correctness and security constraints. The sliced CCR extension is proved in Appendix B.

A.1 The circuit-level algorithms

We now describe the full garbling scheme for circuits composed of free-XOR gates and gates admitting "good" garbling equations (i.e., 2-input gates under RR21's equation, or 3-input gates under the equation of Section 5).

Construction 5 (The Full Garbling Scheme) *The full garbling scheme for circuit C is defined as follows.*

$\mathsf{Garb}(1^\lambda, C)$:

- *Uniformly sample the global key* $\Delta \xleftarrow{\$} 1\|\{0,1\}^{\lambda-1}$.
- *For each circuit input wire* $i \in \mathsf{in}(C)$, *sample its zero label and permute bit* $L_i^0 \xleftarrow{\$} 0\|\{0,1\}^{\lambda-1}, \pi_i \xleftarrow{\$} \{0,1\}$.
- *Traverse the circuit in topological order. For each gate* i*: if it is a free-XOR gate, set* $L_i^0 = \bigoplus_{j\in\mathsf{in}(i)} L_j^0$ *and* $\pi_i = \bigoplus_{j\in\mathsf{in}(i)} \pi_j$. *Otherwise, sample the permute bit uniformly and compute the zero label* L_i^0 *according to fig. 1 with the appropriate garbling equation.*
- *Output* $(\hat{C}, e, d)$, *where* $\hat{C}$ *is the collection of all gates' garbled tables,* e *contains the pairs* $(L_i^0, L_i^0 \oplus \Delta)$ *and permute bits* π_i *for all circuit input wires* i, *and* d *is the concatenation of*

$$(d_o^0 := H(L_o^{\pi_o}; o), d_o^1 := H(L_o^{\pi_o} \oplus \Delta; o))$$

over all circuit output wires o.

$\mathsf{Enc}(e, x)$: *For each circuit input wire* $i \in \mathsf{in}(C)$, *output one of its two labels* $L_i^0 \oplus (x_i + \pi_i)\Delta$, *according to the input bit* x_i.

$\mathsf{Eval}(\hat{C}, X)$: *Starting from the input wire labels, traverse* C *in topological order. For each free-XOR gate, xor its input labels. For other gates, evaluate as specified in eq.* (7). *Collect the computed labels* Y_o *for all circuit output wires* o, *and output* $Y = \{Y_o\}$.

$\mathsf{Dec}(d, Y)$: *Determine the value of each circuit output wire* o *by*

$$\begin{cases} 0 & \text{if } H(Y_o; o) \text{ equals } d_o^0, \\ 1 & \text{if } H(Y_o; o) \text{ equals } d_o^1, \\ \bot & \text{otherwise.} \end{cases}$$

A.2 Correctness

Definition 2 (Correctness). *A garbling scheme is* correct *if for any circuit C and input x,*

$$\Pr_{(\hat{C},e,d)\leftarrow\mathsf{Garb}(1^\lambda,C)}\Big[\mathsf{Dec}(d,\mathsf{Eval}(\hat{C},\mathsf{Enc}(e,x)))=C(x)\Big]\geq 1-\mathrm{negl}(\lambda).$$

Theorem 6. *Our garbling scheme is* correct *if H is a random oracle or satisfies the sliced CCR condition of Section 4.5.*

Proof. By correctness of individual gates, for every wire i, Eval computes the label $L_i^0 \oplus (x_i \oplus \pi_i)\Delta$, where x_i is the wire's value under $C(x)$. For each output wire o with value y_o, we have $Y_o = L_o^0 \oplus (y_o \oplus \pi_o)\Delta$; Dec thus recovers y_o unless the two decoding tags collide. This has probability $2^{-\lambda}$ per output in the random-oracle model and negligible probability under CCR, as shown in Appendix B.4. A union bound over the output wires completes the proof.

A.3 Obliviousness

Definition 3 (Obliviousness). *A garbling scheme is* oblivious *if there exists a simulator $\mathsf{Sim}_{\mathrm{obv}}$ such that for any circuit C and input x, the following two distributions are computationally indistinguishable:*

$$\{(\hat{C},X)\mid(\hat{C},e,d)\leftarrow\mathsf{Garb}(1^\lambda,C),X=\mathsf{Enc}(e,x)\}$$
$$\textit{and }\{(\hat{C},X)\mid(\hat{C},X)\leftarrow\mathsf{Sim}_{\mathrm{obv}}(1^\lambda,\Phi(C))\},$$

Here $\Phi(C)$ specifies the circuit topology and identifies the free-XOR gates.

Theorem 7. *Our garbling scheme is* oblivious *if H is a random oracle.*

Proof. The proof proceeds via a sequence of hybrids from the real world to the evaluator's simulated view. We assume without loss of generality that C consists entirely of 3-input 1-output gates. For each wire i, let x_i denote its value under $C(x)$; the evaluator sees the color bit $\tilde{x}_i := x_i + \pi_i$ and label $L_i^0 \oplus \tilde{x}_i\Delta$.

Hybrid 0 (Real World). $(\hat{C},X)$ is generated by $(\hat{C},e,d)\leftarrow\mathsf{Garb}(1^\lambda,C), X\leftarrow\mathsf{Enc}(e,x)$ as described in 5.

Hybrid 1. The global key, labels, and permute bits for input wires are sampled as in $\mathsf{Garb}(1^\lambda,C)$. For each $i\notin\mathsf{in}(C)$ in topological order with $\mathsf{in}(i)=\{j,k,l\}$ $(j<k<l<i)$:

- If the i-th gate g_i is free-XOR, set $L_i^0 = L_j^0 \oplus L_k^0 \oplus L_l^0$ and $\pi_i = \pi_j + \pi_k + \pi_l$.
- Otherwise, sample $(\mathbf{G},\mathbf{Z})$ and π_i uniformly. Choose a random solution $\mathbf{R}$ of (5) for the gate function $\tilde{g}_i$ (permuted by π_i,π_j,π_k,π_l) satisfying

$$\mathbf{R}(\tilde{x}_j,\tilde{x}_k,\tilde{x}_l) = Z_{\tilde{x}_j,\tilde{x}_k,\tilde{x}_l} \oplus H\left((L_j^{\tilde{x}_j},L_k^{\tilde{x}_k},L_l^{\tilde{x}_l});i\right).$$

  Program the random oracle such that

- for each $(y_j, y_k, y_l) \neq (\tilde{x}_j, \tilde{x}_k, \tilde{x}_l)$,

$$H\left((L_j^{y_j}, L_k^{y_k}, L_l^{y_l}); i\right) = \mathbf{R}(y_j, y_k, y_l) \oplus Z_{y_j, y_k, y_l},$$

- the oracle calls in $\boldsymbol{H} \backslash \boldsymbol{H}_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l}$, i.e.,

$$H\left(b_j L_j^{\tilde{x}_j} \oplus b_k L_k^{\tilde{x}_k} \oplus b_l L_l^{\tilde{x}_l} \oplus \Delta; (i, b_j, b_k, b_l)\right)$$

  for the relevant triples $(b_j, b_k, b_l) \in \{0,1\}^3$, are modified so that the garbling equation (4) holds. If the modification is not unique, pick one at random.

The label L_i^0 is computed from the garbling equation as before.

By uniformity of $(\mathbf{G}, \mathbf{Z})$ and the security constraint that $\boldsymbol{H} \backslash \boldsymbol{H}_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l}$ maps linearly and surjectively onto $\mathbf{G}$, the programmed H remains random. $\mathbf{R}$ is uniformly distributed in the solution space of $\tilde{g}_i$, since its marginal $\mathbf{R}(\tilde{x}_j, \tilde{x}_k, \tilde{x}_l)$ is masked by the uniform $Z_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l}$. The only difference from the previous hybrid is the order in which $(\mathbf{G}, \mathbf{Z})$ and $\boldsymbol{H}$ are sampled; the distribution is identical.

Hybrid 2. Observe that Δ is used only to program the oracle, and the active labels $L_i^{\tilde{x}_i}$ can be computed by simulating the evaluator without knowledge of Δ. Hence the evaluator's view can be generated before programming H:

1. Sample $\pi_i \xleftarrow{\$} \{0,1\}, L_i^{\tilde{x}_i} \xleftarrow{\$} (x_i + \pi_i) \| \{0,1\}^{\lambda-1}$ for all input wires $i \in \mathsf{in}(C)$.
2. Sample $(\mathbf{G}, \mathbf{Z})$ and π_i uniformly for each non-free-XOR gate i.
3. Compute $L_i^{\tilde{x}_i}$ for all $i \notin \mathsf{in}(C)$ by following the evaluator's procedure.

Finally, sample $\Delta \xleftarrow{\$} 1 \| \{0,1\}^{\lambda-1}$ and program H as in **Hybrid 1**. Since only the sampling of Δ and the programming of H are deferred, the distribution of the evaluator's view is unchanged.

Hybrid 3 (Ideal World). The simulator $\mathsf{Sim}_{\mathrm{obv}}$ samples directly:

1. $\tilde{x}_i \xleftarrow{\$} \{0,1\}, L_i^{\tilde{x}_i} \xleftarrow{\$} \tilde{x}_i \| \{0,1\}^{\lambda-1}$ for all $i \in \mathsf{in}(C)$,
2. a uniform garbled table $(\mathbf{G}, \mathbf{Z})$ for each non-free-XOR gate.

In this hybrid, H is not programmed.

Before H is programmed, the evaluator's view is identically distributed as in **Hybrid 2**. Since Δ is independent of the evaluator's view and all modified oracle positions differ by Δ from information present in the view, even an unbounded adversary can distinguish this hybrid from the previous one with probability at most $O(Q \cdot 2^{-\lambda})$, where Q is the number of H-queries.

A.4 Privacy

Definition 4 (Privacy). *A garbling scheme is* private *if there exists a simulator* $\mathsf{Sim}_{\text{pri}}$ *such that for any circuit* C *and input* x*, the following two distributions are computationally indistinguishable:*

$$\{(\hat{C}, X, d) \mid (\hat{C}, e, d) \leftarrow \mathsf{Garb}(1^\lambda, C), X = \mathsf{Enc}(e, x)\}$$
$$\{(\hat{C}, X, d) \mid (\hat{C}, X, d) \leftarrow \mathsf{Sim}_{\text{pri}}(1^\lambda, \Phi(C), C(x))\}.$$

Theorem 8. *Our garbling scheme is* private *if* H *is a random oracle.*

Proof. The simulator $\mathsf{Sim}_{\text{pri}}$ first invokes $\mathsf{Sim}_{\text{obv}}$ to obtain simulated $(\hat{C}, X)$, then runs $\mathsf{Eval}(\hat{C}, X)$ to obtain simulated Y. Finally, it simulates d by assigning the hash of each active label to its semantic output value. For example, if output wire o has value 0 under $C(x)$, set $d_o^0 = H(Y_o; o)$ and draw $d_o^1 \xleftarrow{\$} \{0,1\}^\lambda$; the case $y_o = 1$ is symmetric.

By *obliviousness*, the simulated Y is indistinguishable from real output labels; hence the simulated $d_o^{y_o}$ appears to be generated by Garb. Conversely, without knowledge of Δ, the real-world $d_o^{1-y_o}$ is uniformly random from the adversary's perspective, by the random oracle property.

A.5 Authenticity

Definition 5 (Authenticity). *A garbling scheme is* authentic *if for any circuit* C*, input* x*, and PPT adversary* $\mathcal{A}$*,*

$$\Pr_{(\hat{C}, e, d) \leftarrow \mathsf{Garb}(1^\lambda, C)} \Big[\mathsf{Dec}(d, \mathcal{A}(\hat{C}, \mathsf{Enc}(e, x), d)) \notin \{C(x), \perp\}\Big] \leq \text{negl}(\lambda).$$

Theorem 9. *Our garbling scheme is* authentic *if* H *is a random oracle.*

Proof. By *privacy*, $\mathcal{A}(\hat{C}, \mathsf{Enc}(e, x), d)$ can be replaced by a simulated view via a standard hybrid:

$$\mathcal{A}(\mathsf{Sim}_{\text{pri}}(1^\lambda, \Phi(C), C(x))).$$

For Dec to output a value outside $\{C(x), \perp\}$, there must exist an output wire o with value y_o under $C(x)$ for which $\mathcal{A}$ produces Y_o satisfying $H(Y_o; o) = d_o^{1-y_o}$. Since $d_o^{1-y_o}$ is fresh randomness in $\mathsf{Sim}_{\text{pri}}$, this occurs with negligible probability.

B Security under the CCR Assumption

We use the sliced CCR condition from Section 4.5 and the uncompressed control tables of Figure 1 to prove obliviousness, privacy, and authenticity. We first establish obliviousness and then include the output-decoding hashes in the same argument. For two slices, abbreviate the oracle in (8) as $\mathrm{CCR}_{\Delta}(L;\tau;b_L,b_R)$. Unprimed hybrid numbers refer to the random-oracle proof in Appendix A. We also retain its wire notation: x_i is the semantic value on wire i, π_i is its permutation bit, and $\tilde{x}_i = x_i + \pi_i$ is its color.

B.1 The sliced CCR condition

Definition 6 (Circular Correlation Robustness for s Slices). *Let s divide the label length λ, write $m = \lambda/s$, and let $H : \{0,1\}^{\lambda} \times \mathcal{T} \to \{0,1\}^{m}$ be a hash function, where $\mathcal{T}$ is the set of domain-separation tweaks. Sample Δ as in the garbling scheme: its designated color bit is one and its remaining bits are uniform. Partition it into s slices $\Delta_1,\ldots,\Delta_s \in \{0,1\}^m$. For $L \in \{0,1\}^{\lambda}$, $\tau \in \mathcal{T}$, and $b = (b_1,\ldots,b_s) \in \{0,1\}^s$, use the oracle in (8). We say that H satisfies* CCR for s slices *if, for any non-uniform polynomial-time distinguisher $\mathcal{A}$,*

$$\left|\Pr_{\Delta}[\mathcal{A}^{\mathrm{CCR}_{\Delta}^{(s)}(\cdot)}(1^{\lambda})=1] - \Pr_{\mathcal{R}}[\mathcal{A}^{\mathcal{R}(\cdot)}(1^{\lambda})=1]\right|$$

is negligible. Here $\mathcal{R}$ is a uniformly random function with the same domain and range as $\mathrm{CCR}_{\Delta}^{(s)}$, and $\mathcal{A}$ may query each pair (L,τ) at most once, even with different choices of b.

For $s = 2$, the mask in (8) is precisely $b_L\Delta_L \oplus b_R\Delta_R$. If the desired security parameter is not divisible by s, the definition uses the actual padded label length, with all non-color bits sampled uniformly. For constructions over extension fields, the condition can be applied after expansion into binary coordinates; here s counts the binary slices. Tweaks distinguish the gate, the hash-call index and purpose, and, when needed, the binary output coordinate. Repeated uses of the same hash output reuse its oracle response.

Security constraint II lets us absorb each occurrence of the global offset into an unknown hash output. With s slices, this requires exactly the s-term mask allowed in (8); the garbling equations are unchanged. Appendix B proves obliviousness, privacy, and authenticity for the process in Section 4.3 with any number of slices. Its λ-bit output-decoding hashes are implemented by concatenating s domain-separated calls to H.

B.2 Obliviousness with two slices

We adapt the obliviousness proof of Appendix A to the CCR setting with $s = 2$, $m = \lambda/2$, and gate tweaks in $[|C|] \times \{0,1\}^5$. These tweaks are embedded in $\mathcal{T}$ with a gate-hash domain tag; separate domains are reserved for output decoding and the verification queries used in the proof below. Rather than programming

the random oracle, we route every occurrence of Δ through CCR oracle calls. For convenience, define the helper functions

$$\begin{aligned} H_3 : \left(\{0,1\}^\lambda\right)^3 \times [|C|] &\to \{0,1\}^{\lambda/2} \\ \left((L_j^x, L_k^y, L_l^z); i\right) &\mapsto H(L_j^x; (i,x,y,z,0,0)) \\ &\quad \oplus H(L_k^y; (i,x,y,z,0,1)) \\ &\quad \oplus H(L_l^z; (i,x,y,z,1,0)) \\ H_1 : \{0,1\}^\lambda \times \left([|C|] \times \{0,1\}^3\right) &\to \{0,1\}^{\lambda/2} \\ (L; (i,b_1,b_2,b_3)) &\mapsto H(L; (i,b_1,b_2,b_3,1,1)), \end{aligned}$$

where x, y, z are the least significant bits of L_j^x, L_k^y, L_l^z as before.

Hybrid 0′ (Real World). Identical to **Hybrid 0**, with $H((L_1, L_2, L_3); i)$ calling $H_3(\cdot)$ and $H(L; (i, b_1, b_2, b_3))$ calling $H_1(\cdot)$.

Hybrid 1′. Let $\mathrm{CCR}_\Delta(\cdot)$ be the CCR oracle instantiated with $\Delta \xleftarrow{\$} 1\|\{0,1\}^{\lambda-1}$. For all $i \in \mathsf{in}(C)$, sample $\pi_i \xleftarrow{\$} \{0,1\}, L_i^{\tilde{x}_i} \xleftarrow{\$} (x_i + \pi_i)\|\{0,1\}^{\lambda-1}$. For each $i \notin \mathsf{in}(C)$ in topological order with $\mathsf{in}(i) = \{j,k,l\}$: if gate i is free-XOR, set $L_i^{\tilde{x}_i} = L_j^{\tilde{x}_j} \oplus L_k^{\tilde{x}_k} \oplus L_l^{\tilde{x}_l}$ and $\pi_i = \pi_j + \pi_k + \pi_l$. Otherwise:

- Sample $Z_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l}$ uniformly, then sample a solution $\mathbf{R}$ of (5) for $\tilde{g}_i$ (permuted by $\pi_i, \pi_j, \pi_k, \pi_l$) satisfying

  $$\mathbf{R}(\tilde{x}_j, \tilde{x}_k, \tilde{x}_l) = Z_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l} \oplus H_3\left((L_j^{\tilde{x}_j}, L_k^{\tilde{x}_k}, L_l^{\tilde{x}_l}); i\right).$$

  For all $(y_j, y_k, y_l) \neq (\tilde{x}_j, \tilde{x}_k, \tilde{x}_l)$, generate

  $$\begin{aligned} Z_{y_j,y_k,y_l} &= \mathbf{R}(y_j, y_k, y_l) \oplus H_3\left(L_j^{y_j}, L_k^{y_k}, L_l^{y_l}; i\right) \\ &= \mathbf{R}(y_j, y_k, y_l) \oplus H\left(L_j^{y_j}; (i, y_j, y_k, y_l, 0, 0)\right) \\ &\quad \oplus H\left(L_k^{y_k}; (i, y_j, y_k, y_l, 0, 1)\right) \oplus H\left(L_l^{y_l}; (i, y_j, y_k, y_l, 1, 0)\right). \end{aligned}$$

  If $y_j \neq \tilde{x}_j$, obtain

  $$H\left(L_j^{y_j}; (i, y_j, y_k, y_l, 0, 0)\right) = \mathrm{CCR}_\Delta\left(L_j^{\tilde{x}_j}; (i, y_j, y_k, y_l, 0, 0); 0, 0\right)$$

  via the CCR oracle; y_k, y_l are treated similarly.
- Compute $\mathbf{G}$ as in the real world by

  $$\mathbf{G} = [\mathbf{C} \mid \mathbf{D}] \begin{bmatrix} \boldsymbol{H} \backslash \boldsymbol{H}_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l} \\ \Delta \end{bmatrix} \oplus \mathrm{rest},$$

  where $\mathbf{C}$ is the submatrix of $\mathbf{V}^{-1}\mathbf{M}$ obtained by removing the first two rows and all columns indexed by $\boldsymbol{H}_{\tilde{x}_j, \tilde{x}_k, \tilde{x}_l}$, and $\mathbf{D}$ collects all coefficients of $\Delta = (\Delta_L, \Delta_R)$; all terms in "rest" are computable from the active labels alone. By

security constraint II, $\mathbf{C}$ has full row rank. Without loss of generality, assume its rightmost columns are linearly independent, so that for some invertible $\mathbf{C}'$,

$$\mathbf{C} = \mathbf{C}' \begin{bmatrix} * & \cdots & * & 1 \\ \vdots & & \vdots & & \ddots \\ * & \cdots & * & & 1 \end{bmatrix}, \qquad [\mathbf{C} \mid \mathbf{D}] = \mathbf{C}' \left[\begin{array}{ccccc|c} * & \cdots & * & 1 & & \\ \vdots & & \vdots & & \ddots & \\ * & \cdots & * & & 1 & \end{array} \right. \left. \mathbf{C}'^{-1}\mathbf{D} \right].$$

Partition $\boldsymbol{H}\backslash\boldsymbol{H}_{\tilde{x}_j,\tilde{x}_k,\tilde{x}_l}$ as $\begin{bmatrix} \boldsymbol{H}_* \\ \boldsymbol{H}_{\mathbf{I}} \end{bmatrix}$ so that

$$\begin{aligned} \mathbf{C}\left(\boldsymbol{H}\backslash\boldsymbol{H}_{\tilde{x}_j,\tilde{x}_k,\tilde{x}_l}\right) &= \mathbf{C}' \begin{bmatrix} * & \cdots & * & 1 \\ \vdots & & \vdots & & \ddots \\ * & \cdots & * & & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_* \\ \boldsymbol{H}_{\mathbf{I}} \end{bmatrix} \\ &= \mathbf{C}' \left(\begin{bmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{bmatrix} \boldsymbol{H}_* \oplus \boldsymbol{H}_{\mathbf{I}} \right). \end{aligned}$$

Then

$$[\mathbf{C} \mid \mathbf{D}] \begin{bmatrix} \boldsymbol{H}\backslash\boldsymbol{H}_{\tilde{x}_j,\tilde{x}_k,\tilde{x}_l} \\ \Delta \end{bmatrix} = \mathbf{C}' \left(\begin{bmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{bmatrix} \boldsymbol{H}_* \oplus \boldsymbol{H}_{\mathbf{I}} \oplus (\mathbf{C}'^{-1}\mathbf{D})\Delta \right)$$

can be evaluated using only legal CCR oracle calls:

- Every hash value in $\boldsymbol{H}_*$ has the form

$$H_1(b_j L_j^{\tilde{x}_j} \oplus b_k L_k^{\tilde{x}_k} \oplus b_l L_l^{\tilde{x}_l} \oplus \Delta; (i, b_j, b_k, b_l))$$

  for some $(b_j, b_k, b_l) \in \{0,1\}^3$, and can be obtained via

$$\mathrm{CCR}_{\Delta}\left(b_j L_j^{\tilde{x}_j} \oplus b_k L_k^{\tilde{x}_k} \oplus b_l L_l^{\tilde{x}_l}; (i, b_j, b_k, b_l, 1, 1); 0, 0\right).$$

- Each row of $\boldsymbol{H}_{\mathbf{I}} \oplus (\mathbf{C}'^{-1}\mathbf{D})\Delta$ is of the form

$$H_1\left(b_j L_j^{\tilde{x}_j} \oplus b_k L_k^{\tilde{x}_k} \oplus b_l L_l^{\tilde{x}_l} \oplus \Delta; (i, b_j, b_k, b_l)\right) \oplus b_L \Delta_L \oplus b_R \Delta_R,$$

  where (b_L, b_R) is the corresponding row of $\mathbf{C}'^{-1}\mathbf{D}$. This equals

$$\mathrm{CCR}_{\Delta}\big(b_j L_j^{\tilde{x}_j} \oplus b_k L_k^{\tilde{x}_k} \oplus b_l L_l^{\tilde{x}_l}; (i, b_j, b_k, b_l, 1, 1); b_L, b_R\big).$$

After generating $(\mathbf{G}, \mathbf{Z})$, the label $L_i^{\tilde{x}_i}$ is computed from the active labels, following the evaluator's procedure.

This hybrid merely reroutes all uses of Δ through the CCR oracle; the distribution is unchanged.

Hybrid 2′. Replace the outputs of $\mathrm{CCR}_{\Delta}(\cdot)$ with independent random values. Then $(\mathbf{G}, \mathbf{Z})$ for each non-free-XOR gate is uniform, because:

- $Z_{\tilde{x}_j,\tilde{x}_k,\tilde{x}_l}$ is sampled uniformly,
- every Z_{y_j,y_k,y_l} with $(y_j, y_k, y_l) \neq (\tilde{x}_j, \tilde{x}_k, \tilde{x}_l)$ is masked by a disjoint CCR output,
- $\mathbf{G}$ is masked by $\mathbf{C}'\big(\boldsymbol{H}_{\mathbf{I}} + \mathbf{C}'^{-1}\mathbf{D}\Delta\big)$. Since the rows of $\boldsymbol{H}_{\mathbf{I}} + \mathbf{C}'^{-1}\mathbf{D}\Delta$ are independent CCR outputs and $\mathbf{C}'$ is invertible, the rows of $\mathbf{G}$ are independently uniform.

By the CCR property of H, this hybrid is indistinguishable from **Hybrid 1′**. Since the distribution matches **Hybrid 3**, this completes the proof of *obliviousness* under the CCR assumption.

B.3 Extension to s slices

The step that absorbs Δ into the unknown hash outputs does not depend on having two slices. For s slices, the matrix $\mathbf{D}$ has s columns, one for each slice of Δ. Each row of $\mathbf{C}'^{-1}\mathbf{D}$ is then a vector $b \in \{0,1\}^s$, so its contribution to the corresponding unknown hash output is exactly the mask $\bigoplus_{a=1}^{s} b_a \Delta_a$ allowed by (8). Security constraint II supplies the same full-row-rank property of $\mathbf{C}$, and hence the same invertible change of basis $\mathbf{C}'$. Using the tweaks specified above and reusing each response, the CCR queries obey the same one-query-per-pair restriction. Thus the linear masking argument extends to any number of slices; only the dimension of the mask vector changes.

B.4 Privacy and authenticity

We instantiate the λ-bit output-decoding hash $H(L; o)$ in Construction 5 as

$$H_{\mathrm{out}}(L; o) = \Big\|_{a=1}^{s} H(L; (\mathsf{out}, o, a)).$$

Let $y_o = C(x)_o$ be the semantic value of output wire o, and let Y_o be its active label. The two decoding tags are

$$d_o^{y_o} = H_{\mathrm{out}}(Y_o; o), \qquad d_o^{1-y_o} = H_{\mathrm{out}}(Y_o \oplus \Delta; o).$$

Privacy. Extend **Hybrid 1′** by computing the active tag directly and obtaining every slice of the inactive tag from

$$\mathrm{CCR}_{\Delta}^{(s)}(Y_o; (\mathsf{out}, o, a); 0^s).$$

The reduction knows Y_o from evaluation, and the fresh tweaks make these queries legal. Replacing these replies together with the gate-table replies in **Hybrid 2′** makes the inactive tags independently uniform. The resulting joint view is exactly the privacy simulation of Appendix A, using only $\Phi(C)$ and $C(x)$.

Authenticity. We first show that CCR prevents finding a preimage of a uniform decoding tag. Fix an output wire o and write $f(L) = H_{\mathrm{out}}(L; o)$. Suppose an efficient algorithm I finds a preimage of a uniform $U \in \{0,1\}^\lambda$ with probability ε. Since the domain and range of f have the same size, grouping inputs by their images gives

$$\Pr_{X \xleftarrow{\$} \{0,1\}^\lambda}[I(f(X)) = X] = 2^{-\lambda} \sum_{u \in \{0,1\}^\lambda} \Pr[f(I(u)) = u] = \varepsilon. \tag{24}$$

The probabilities include the randomness of I. This identity allows collisions in f and any efficiently sampled auxiliary input independent of X, by first fixing that input.

A CCR distinguisher now privately samples $L \xleftarrow{\$} \{0,1\}^\lambda$ and uses s zero-mask queries to obtain $U = f(L \oplus \Delta)$ in the real experiment. It gives only U, not L, to I, and sets $\Delta' = I(U) \oplus L$. Since $L \oplus \Delta$ is uniform over the full label space, (24) gives $\Pr[\Delta' = \Delta] = \varepsilon$. The distinguisher checks its guess using fresh verification tweaks $\nu_1, \ldots, \nu_s$, accepting if

$$\mathop{\Big\|}_{a=1}^{s} \mathrm{CCR}_{\Delta}^{(s)}(0^\lambda; \nu_a; 0^s) = \mathop{\Big\|}_{a=1}^{s} H(\Delta'; \nu_a).$$

The real experiment accepts with probability at least ε; the ideal experiment accepts with probability $2^{-\lambda}$, since its fresh replies are uniform. All queries use distinct tweaks. Hence sliced CCR makes ε negligible without an additional preimage-resistance assumption.

Finally, replace the adversary's garbling view by the joint privacy simulation above. Any accepted incorrect output must contain a label W_o satisfying $H_{\mathrm{out}}(W_o; o) = d_o^{1-y_o}$ for some output wire o. Placing a uniform challenge in a randomly chosen inactive tag and simulating the remaining view gives a preimage algorithm, losing at most a factor equal to the number of outputs. Its auxiliary view is sampled independently of Δ and the hidden L, so the preceding argument proves authenticity.

Each inactive tag in the ideal view collides with its active counterpart with probability $2^{-\lambda}$. A union bound and the same CCR hybrid therefore give a negligible decoding-error probability.

C Proof of the Framework Lower Bound

We prove Theorem 2 using the stacked matrices of the framework. Throughout, the coefficient field is $\mathbb{F}_2$, $n \geq 2$, $s \geq 1$, $N = 2^n$, and $d = s + r$. The formal input variables are $x = (x_1, \ldots, x_n)$, and $p = (p_1, \ldots, p_n)$ denotes an evaluator's color vector. The matrix $\mathbf{V} \in \mathbb{F}_2^{sN \times d}$ has independent columns and consists of the blocks

$$\mathbf{V}(x) = [\, \mathbf{I}_s \ \mathbf{A}(x) \,], \qquad x \in \{0,1\}^n.$$

Thus its first s columns are the constant output-label coordinates, and its remaining r columns are ciphertext coordinates. Define the projection onto the latter by $\boldsymbol{\Pi} = [\, \mathbf{0}_{r \times s} \ \mathbf{I}_r \,]$.

Correctness and local control freedom. Let $\mathbf{X}_i$ be the block-diagonal matrix whose block at x is $x_i \mathbf{I}_s$, and put

$$\mathbf{T} = [\, \mathbf{V} \ \mathbf{X}_1\mathbf{V} \ \cdots \ \mathbf{X}_n\mathbf{V} \,], \qquad \mathcal{K} = \ker \mathbf{T}.$$

The unknown in this linear system consists of $n+1$ coefficient vectors $q_0, \ldots, q_n \in \mathbb{F}_2^d$. Correctness for all gate truth tables requires $\operatorname{rank} \mathbf{T} = sN$: the right-hand sides obtained by varying a scalar truth table and its output coordinate span $\mathbb{F}_2^{sN}$. Rank–nullity therefore gives

$$\dim \mathcal{K} = (n+1)(s+r) - sN. \tag{25}$$

For each fixed p, define the ns-by-$(n+1)d$ matrix

$$\mathbf{E}_p = [\, \mathbf{0}_{ns \times d} \ \ \mathbf{I}_n \otimes \mathbf{V}(p) \,].$$

Here the second block is block diagonal, with n copies of $\mathbf{V}(p)$. In particular,

$$\mathbf{E}_p \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} \mathbf{V}(p) q_1 \\ \vdots \\ \mathbf{V}(p) q_n \end{bmatrix}.$$

These are the local input-label coefficients for one column of the control matrices. The constraints act separately on their s columns, so this suffices to express S1. For a fixed right-hand side, the solutions of the correctness equations form an affine translate of $\mathcal{K}$. A uniform solution induces a uniform distribution on the corresponding affine translate of $\mathbf{E}_p\mathcal{K}$. Consequently the joint local controls are uniform over all $\mathbb{F}_2^{ns}$ precisely when

$$\operatorname{rank}(\mathbf{E}_p|_{\mathcal{K}}) = ns. \tag{26}$$

The coefficient vector q_0 introduces no extra sampling choice: once the other vectors satisfy correctness, it is unique because $\mathbf{V}$ has independent columns.

The supported columns required by S2. Allow an arbitrary number t of main-table hash pairs. Pair j has a nonzero binary linear form $\ell_j(x) = \sum_i \ell_{j,i} x_i$. Its two coefficient columns $\mathbf{M}_{j,b}$, indexed by $b \in \{0,1\}$, belong to colspace($\mathbf{V}$), so they have unique representations

$$\mathbf{M}_{j,b} = \mathbf{V} a_{j,b}, \qquad a_{j,b} \in \mathbb{F}_2^d.$$

The support requirement is

$$\mathbf{V}(x) a_{j,b} = 0 \qquad \text{whenever } \ell_j(x) \neq b. \tag{27}$$

At color p, the unavailable side is $1 - \ell_j(p)$. S2 is therefore exactly the rank requirement

$$\operatorname{rank}[\, \boldsymbol{\Pi} a_{j,1-\ell_j(p)} \,]_{j=1}^{t} = r \qquad \text{for every } p \in \{0,1\}^n. \tag{28}$$

There is no assumption that $t = r$. Labels and columns may repeat, and zero columns are allowed. A column without a paired partner can be completed by a zero column on the other side. Nonconstant affine labels are also covered, by absorbing their constant terms into the side index b.

Averaging the ranks of support relations. For every supported column, form the coefficient vector

$$z_{j,b} = \begin{bmatrix} b a_{j,b} \\ \ell_{j,1} a_{j,b} \\ \vdots \\ \ell_{j,n} a_{j,b} \end{bmatrix}.$$

It lies in $\mathcal{K}$. Indeed, the block of $\mathbf{T} z_{j,b}$ at x is

$$(b + \ell_j(x)) \mathbf{V}(x) a_{j,b} = 0 :$$

on side b the scalar factor is zero, and on the other side the coefficient column vanishes by (27).

Let $\mathcal{Z}$ be the span of all the $z_{j,b}$, and set $k = \dim \mathcal{Z}$. Extract a basis from these original generating vectors and place its columns in a matrix $\mathbf{Z}$. This basis need not contain both sides of any pair. Define

$$\mathbf{L}_p = [\, \boldsymbol{\Pi} \;\; p_1 \boldsymbol{\Pi} \;\; \cdots \;\; p_n \boldsymbol{\Pi} \,].$$

This matrix combines coefficient blocks; it does not evaluate their tables at p. On a support relation its action is

$$\mathbf{L}_p z_{j,b} = (b + \ell_j(p)) \boldsymbol{\Pi} a_{j,b}.$$

The known-side relations have zero image, while the unavailable-side relations give the columns in (28). Since $\mathbf{Z}$ spans all the relations, S2 implies $\operatorname{rank}(\mathbf{L}_p \mathbf{Z}) = r$ for every p.

Write (ℓ_a, b_a) for the original label and side of the a-th basis column, for $1 \le a \le k$. That column has zero image whenever $\ell_a(p) + b_a = 0$. Hence

$$r \le \#\{a : \ell_a(p) + b_a = 1\}.$$

Every nonzero binary linear form takes each value at exactly half the inputs. Averaging this inequality over all N colors gives

$$r \le \frac{1}{N} \sum_{p \in \{0,1\}^n} \#\{a : \ell_a(p) + b_a = 1\} = \frac{k}{2}.$$

Thus $\dim \mathcal{K} \ge \dim \mathcal{Z} \ge 2r$. Together with (25), this yields

$$(n+1)(s+r) - sN \ge 2r, \qquad (n-1)r \ge s(N-n-1).$$

This first inequality uses C and S2, but not S1. The extracted basis is a basis of coefficient relations in $\mathcal{K}$; it is not a common selection of r hash pairs for every evaluator color. In particular, the argument does not discard redundant pairs from the S2 witness.

Preserving the local controls. Fix p. Let $\mathbf{Z}_p$ contain the relations $z_{j,1-\ell_j(p)}$ from all unavailable sides. Their columns belong to $\mathcal{K}$, and (28) gives

$$\operatorname{rank}(\mathbf{L}_p \mathbf{Z}_p) = r, \qquad \operatorname{rank} \mathbf{Z}_p \ge r.$$

Every unavailable supported column vanishes at p, so each of the last n coefficient blocks of its relation evaluates to zero there. Consequently

$$\mathbf{E}_p \mathbf{Z}_p = 0, \qquad \operatorname{colspace}(\mathbf{Z}_p) \subseteq \ker(\mathbf{E}_p|_{\mathcal{K}}).$$

By S1 and rank–nullity,

$$\dim \mathcal{K} = \operatorname{rank}(\mathbf{E}_p|_{\mathcal{K}}) + \dim \ker(\mathbf{E}_p|_{\mathcal{K}}) \ge ns + r.$$

Substituting (25) now gives

$$(n+1)(s+r) - sN \ge ns + r, \qquad nr \ge s(N-1).$$

This second inequality uses C, S1, and S2. Since r is an integer, the two inequalities imply

$$r \ge \max\left\{ \left\lceil \frac{s(2^n - n - 1)}{n-1} \right\rceil, \left\lceil \frac{s(2^n - 1)}{n} \right\rceil \right\}.$$

Dividing by s proves the claimed ratio bound

$$\frac{r}{s} \ge \max\left\{ \frac{2^n - n - 1}{n-1}, \frac{2^n - 1}{n} \right\}.$$

The proof permits arbitrary redundant supported-column witnesses and assumes neither a minimal paired witness nor a DAG presentation. It does use the balanced affine color classes that determine main-table hash availability: arbitrary nonlinear availability conditions are outside the theorem. The conclusion concerns the framework's term proportional to the label length, before control information or slice padding.

D Randomized Search and Verification

The pruning idea is simple: reject a partial construction when the remaining columns cannot fill its rank deficit. The tests below make this precise for correctness and for the local control freedom required by S1. S2 follows from the triangular DAG recurrence once the evaluation columns are independent.

Fix the number of inputs n, a coefficient field K of characteristic two, s label slices, and a target of r ciphertext slices. Put $N = 2^n$. All dimensions below are over K; the four-input sampler uses $K = \mathbb{F}_4$ and $(s, r) = (4, 15)$. As in Section 5.1, $x = (x_1, \ldots, x_n)$ denotes the formal input colors, while $p \in \{0,1\}^n$ is an evaluator's fixed color vector.

What the ranks measure. After j nodes, let $\mathbf{A}_j \in K^{Ns\times j}$ stack their columns over all input colors, and write $\mathbf{A}_j(p)$ for its $s \times j$ block at p. With $\mathbf{E} = \mathbf{1}_N \otimes \mathbf{I}_s$, the completed matrix is $\mathbf{V} = [\mathbf{E}\ \mathbf{A}_r]$. Let $\mathbf{X}_i$ be the $Ns \times Ns$ block-diagonal matrix whose block at x is $x_i\mathbf{I}_s$. Correctness asks whether the columns of $\mathbf{V}$, together with their products by individual input bits, span every output truth table:

$$\mathbf{T} = [\mathbf{V}\ \mathbf{X}_1\mathbf{V}\ \cdots\ \mathbf{X}_n\mathbf{V}], \qquad \operatorname{rank}\mathbf{T} = Ns. \tag{29}$$

The complementary hash identities already ensure $\operatorname{colspace}(\mathbf{M}) \subseteq \operatorname{colspace}(\mathbf{V})$.

S1 asks whether changes to a solution can vary all ns local control coefficients without changing correctness. Here we consider one column of each control matrix; their columns obey separate equations. The local coefficients are obtained by evaluating these columns at p.

The implementation uses smaller matrices obtained by removing the constant and linear truth tables, which are already supplied by $\mathbf{E}, \mathbf{X}_1\mathbf{E}, \ldots, \mathbf{X}_n\mathbf{E}$. Choose a full-row-rank matrix $\mathbf{P} \in K^{q_C\times Ns}$, where $q_C = s(N - n - 1)$, with $\ker\mathbf{P} = \operatorname{colspace}[\mathbf{E}\ \mathbf{X}_1\mathbf{E}\ \cdots\ \mathbf{X}_n\mathbf{E}]$. Define

$$\boldsymbol{\Theta}_j = \mathbf{P}[\mathbf{A}_j\ \mathbf{X}_1\mathbf{A}_j\ \cdots\ \mathbf{X}_n\mathbf{A}_j], \qquad \rho_j = \operatorname{rank}\boldsymbol{\Theta}_j. \tag{30}$$

Thus ρ_j counts the non-affine directions supplied so far, and C is equivalent to $\rho_r = q_C$.

For the corresponding local test, let e_i be the i-th input unit vector and put $p^{(i)} = p + e_i$. Define the $s \times j$ difference matrix $\mathbf{U}^{(i)}_{p,j}$ and its associated block by

$$\begin{aligned}\mathbf{U}^{(i)}_{p,j} &= \mathbf{A}_j(p^{(i)}) + \mathbf{A}_j(p),\\ \mathbf{J}^{(i)}_{p,j} &= [\,\mathbf{U}^{(i)}_{p,j} \quad p^{(i)}_1\mathbf{U}^{(i)}_{p,j} \quad \cdots \quad p^{(i)}_n\mathbf{U}^{(i)}_{p,j}\,].\end{aligned}$$

Stack the n blocks $\mathbf{J}^{(i)}_{p,j}$ into the $ns \times (n+1)j$ matrix $\mathbf{J}_{p,j}$, and define

$$\mathbf{B}_{p,j} = \begin{bmatrix}\boldsymbol{\Theta}_j\\ \mathbf{J}_{p,j}\end{bmatrix}, \qquad b_{p,j} = \operatorname{rank}\mathbf{B}_{p,j}. \tag{31}$$

These difference blocks express the same local changes after eliminating the affine coefficients: the linear coefficients of an affine truth table are its differences

between p and $p + e_i$. Consequently $b_{p,j} - \rho_j$ is the dimension of the available local changes. The final tests are therefore

$$\rho_r = s(N - n - 1), \qquad b_{p,r} - \rho_r = ns \quad \text{for every } p,$$

or, equivalently, $\rho_r = s(N - n - 1)$ and $b_{p,r} = s(N - 1)$. Every input is checked; no input sampling is used.

Rejecting prefixes that cannot be completed. A new DAG column contributes at most $n - 1$ to ρ_j and at most n to $b_{p,j}$. The complementary support identities give two relations between its $n + 1$ correctness columns and earlier columns; a linear combination of them also vanishes in the local test. With $r - j$ columns left, a necessary condition for completion is

$$\begin{aligned} \rho_j + (n-1)(r-j) &\geq s(N-n-1), \\ b_{p,j} + n(r-j) &\geq s(N-1) \qquad \text{for every } p. \end{aligned} \tag{32}$$

For example, the three-input target $(n, s, r) = (3, 3, 7)$ needs correctness rank 12 and combined rank 21. After four columns, the three remaining columns can add at most six and nine, respectively. We therefore reject a prefix with $\rho_4 < 6$ or with $b_{p,4} < 12$ at even one input. More generally, the bounds become $2j - \rho_j \leq 2$ and $3j - b_{p,j} \leq 0$.

For the targets satisfying $nr = s(N - 1)$, the code also applies a cheap necessary test at the origin. For each $k \leq j$, stack $\ell_{k,1}c_k, \ldots, \ell_{k,n}c_k$ vertically into one ns-entry column; the resulting j columns must be independent. This test is used for $n = 3, 4$, not for $n = 5, 6, 7$, and does not replace the full C and S1 checks.

Sampling and verification. For $n = 3, 4$, each node proposal independently samples a nonzero binary form uniformly, a uniform constant in K^s, and uniform edge coefficients from every earlier node. Forms may repeat, and zero coefficients are allowed. We reject dependent columns and prefixes failing the tests above, redraw rejected proposals, and restart when a node exhausts its budget of 128 proposals.

For $n = 5, 6, 7$, we instead sample complete DAGs in blocks of widths 2^{13}, $(2, 3, 4^{13})$, and $(2, 2, 4^{29})$, respectively; an exponent denotes repetition. Each block shares one uniform nonzero binary form. Forms are independent across blocks and may repeat. All constant entries and all coefficients from earlier blocks are sampled independently and uniformly; there are no edges within a block. Thus every draw uses fresh coefficients within the fixed profile. Complete draws are screened for column independence and C, then for S1 at every input, stopping at the first failure. These measurements do not use prefix pruning.

Every successful literal construction is checked separately for column independence, C, all S1 ranks, both support identities per node, and every S2 matrix.

The rank thresholds exclude no valid completion within the stated DAG family. However, a finite proposal or restart budget may abandon a viable prefix.

The DAG family, coefficient field, and block profile restrict the search space; general S2 allows supported spanning families without a DAG or a minimal pairing. Rejection and restarting do not sample constructions uniformly, and bounded failure is not an impossibility result.

Search costs. Table 5 reports five runs per arity using the samplers above, with predeclared seeds, fixed sampling configurations, a 180-second search budget, and a cap of one million chains or complete draws per run. Search time is measured to the first candidate passing the search tests and includes rejected proposals; independent full verification is timed separately. The experiments use Python 3.9.6 on an Apple M2 Mac mini with 16 GB of memory, running one search process at a time.

Table 5: Observed search and verification times in seconds. Each row uses five predeclared seeds and a 180-second search budget per run. Search time includes rejected candidates; the final column gives the median time for independent full verification.

n sampler	successes	search median	search range	verification
3 Incremental	5/5	0.486	0.133–1.995	0.001
4 Incremental	5/5	0.515	0.057–2.178	0.024
5 Block draws	5/5	24.963	4.612–160.489	0.034
6 Block draws	5/5	1.011	0.080–5.064	0.388
7 Block draws	5/5	1.306	1.078–4.262	4.437

All five runs succeed for each arity. These are observed outcomes for the specified samplers, not a general success-probability or running-time bound. The wider spread at $n = 5$ illustrates why one successful run would not reliably estimate search cost. The timings exclude the preliminary experiments used to select the sampling configurations. The artifact records those experiments separately, along with commands, seeds, coefficients, and verification results. The $n = 4$ result uses native $\mathbb{F}_4$ coefficients; the binary target $(s, r) = (4, 15)$ remains unresolved.

E Certificates for the Finite Constructions

Proof of Theorem 3.

Proof. For each arity, the search produces literal coefficients for (11). Exact matrix elimination verifies that $\mathbf{V}$ has independent columns, that C holds, and that S1 holds at every input. S2 follows from (12). The larger certificates and their rank checks are summarized below. For $n = 4$, the native construction over $\mathbb{F}_4$ has $(s, r) = (4, 15)$; expansion into binary coordinates gives $(8, 30)$. Substituting the stated parameters into (10) establishes optimality.

For $n \in \{5, 6, 7\}$, the selected search certificates have $s = n - 1$ and $r = 2^n - n - 1$. We specify these instances by their literal coefficient data rather than a rule for generating a family. All search results and verification code will be provided separately. Each certificate lists the binary masks of the forms ℓ_j, the constant vectors c_j, and every nonzero dependency coefficient a_{ij}. Substituting these entries into (11) determines all entries of the evaluation and hash-coefficient matrices.

The search seed is recorded for reproducibility of an experiment; it is not a substitute for a literal certificate. The verifier reads the stored coefficients and reconstructs the matrices independently of the search procedure. It checks column independence, C, S1 at every input, both support identities for each hash pair, and S2 at every input.

The rank matrices below are defined in Appendix D. The exact binary ranks are as follows. Each entry in the last column holds at every one of the 2^n Boolean inputs.

n	$\operatorname{rank} \mathbf{V}$	$\operatorname{rank} \boldsymbol{\Theta}_r$	$\operatorname{rank} \mathbf{B}_{p,r}$
5	30	104	124
6	62	285	315
7	126	720	762

The independently reconstructed matrices also satisfy every support identity and every selected-frame rank test. These exact checks establish the finite instances without requiring a general theorem about the pattern of their coefficients. Search times and certificate verification times are recorded separately.

F Proof of the General Construction

We first verify the matrix construction from a family satisfying (17). We then construct a sufficiently small family from an asymmetric cover. The covering bound is the only external existence result used in this proof.

F.1 The three matrix conditions

Let $\mathcal{F} \subseteq 2^{[n]}$ contain $\varnothing$ and all singletons and satisfy (17). Write $b = |\mathcal{F}|$, let $e_1, \ldots, e_s$ be the standard basis of $\mathbb{F}_2^s$, and form $\mathbf{V}$ by stacking $\mathbf{V}(x) = [m_I(x)\mathbf{I}_s]_{I \in \mathcal{F}}$ over $x \in \{0,1\}^n$. A column is the truth table of $m_I(x)e_a$ for some $I \in \mathcal{F}$ and $a \in [s]$. The uniqueness of multilinear representations of Boolean functions implies that these sb columns are independent. The first s are constant, so there are $r = s(b-1)$ table columns.

Correctness (C). Let $\mathbf{X}_i$ be the diagonal $s2^n$-by-$s2^n$ matrix whose block at input x is $x_i\mathbf{I}_s$. Every vector-valued truth table is a linear combination of the tables

$m_J e_a$, where $J \subseteq [n]$ and $a \in [s]$. If $J = \varnothing$, the table is already a column of $\mathbf{V}$. Otherwise select $i \in J$ with $J \setminus \{i\} \in \mathcal{F}$. Then

$$m_J(x)e_a = x_i m_{J\setminus\{i\}}(x)e_a$$

is a column of $\mathbf{X}_i\mathbf{V}$. Therefore

$$\operatorname{rank}[\,\mathbf{V}\ \mathbf{X}_1\mathbf{V}\ \cdots\ \mathbf{X}_n\mathbf{V}\,] = s2^n,$$

which is precisely the solvability condition for every gate table.

Local freedom (S1). Fix an input $p \in \{0,1\}^n$ and arbitrary prescribed vectors $v_1, \ldots, v_n \in \mathbb{F}_2^s$. With $x \in \{0,1\}^n$ as a formal variable, define

$$f_i(x) = v_i \quad (1 \le i \le n), \qquad f_0(x) = \sum_{i=1}^{n} x_i v_i.$$

The selected constant and singleton columns place every f_i, including f_0, in colspace($\mathbf{V}$). They satisfy

$$f_0(x) + \sum_{i=1}^{n} x_i f_i(x) = 0, \qquad (f_1(p), \ldots, f_n(p)) = (v_1, \ldots, v_n).$$

In matrix terms, their coefficient vectors give an element of $\ker \mathbf{T}$, with $\mathbf{T}$ defined in (29), having any prescribed evaluations in the last n blocks. Thus the kernel-evaluation map is surjective at every p. The correct-control solutions form an affine space, and surjectivity makes every local value have equally many preimages. Thus a uniform solution induces uniform local controls.

To spell out the same argument for full control matrices, let $\mathbf{R}_i(x)$ be the input-label control matrices and write correctness as $\mathbf{R}_0(x) + \sum_i x_i \mathbf{R}_i(x) = \tilde{g}(x)\mathbf{I}_s$, where $\mathbf{R}_0$ is the remainder $\mathbf{R}_\Delta + \mathbf{T}_{\tilde{g}}$ in (5). Starting from any solution, choose arbitrary constant matrices $\mathbf{K}_i \in \mathbb{F}_2^{s\times s}$ and replace

$$\mathbf{R}_i(x) \longmapsto \mathbf{R}_i(x) + \mathbf{K}_i, \qquad \mathbf{R}_0(x) \longmapsto \mathbf{R}_0(x) + \sum_i x_i \mathbf{K}_i.$$

The two added terms cancel in the correctness equation. Since the constant and singleton monomials are selected, every modified table still has its columns in colspace($\mathbf{V}$). At any fixed color p, the matrices $\mathbf{K}_i$ can change all local controls $\mathbf{R}_i(p)$ arbitrarily. A uniform solution therefore gives uniform local controls, independent of the permuted gate table and hence of the permute bits.

Unknown hash values (S2). For each nonempty selected set $I \in \mathcal{F}$, fix $i(I) \in I$ and a selected parent $P(I) = I \setminus \{i(I)\}$. Writing $i = i(I)$, the scalar coefficients form the complementary supported pair

$$\begin{aligned} h_{I,1} &= m_I = x_i m_{I\setminus\{i\}}, \\ h_{I,0} &= m_I + m_{I\setminus\{i\}} = (1 + x_i) m_{I\setminus\{i\}}. \end{aligned} \tag{33}$$

For every $a \in [s]$, multiply the pair by e_a and take one hash slot labeled by the input coordinate $x_{i(I)}$, with coefficient columns

$$h_{I,a,1}(x) = m_I(x)e_a, \qquad h_{I,a,0}(x) = (m_I(x) + m_{P(I)}(x))e_a.$$

The first vanishes when $x_{i(I)} = 0$, and the second when $x_{i(I)} = 1$. Both lie in colspace($\mathbf{V}$). They can be assigned to the two domain-separated oracle values on the labels $A^1_{i(I)}$ and $A^0_{i(I)}$, respectively. The slot index is included in the oracle tweak, so slots with the same input coordinate still use independent oracle values. Their visible-side queries use only input labels and are therefore nonadaptive.

It remains to verify the rank after removing the constant output columns. Let $u_{I,a}$ denote the coordinate vector of the table column $m_I e_a$ modulo constants. At the evaluator input p, the coefficient of the unknown-side oracle value in this slot is

$$u_{I,a} + p_{i(I)} u_{P(I),a},$$

where $u_{\varnothing,a} = 0$. Order nonempty selected sets by cardinality. Every parent precedes its child, so these columns form an upper triangular r-by-r matrix with diagonal entries one. Its rank is r for every p, proving S2. Together with the previous two paragraphs, this proves Proposition 1.

F.2 Hash coefficients for the three-input example

At $n = 3$ we may choose

$$B_{\mathcal{F}} = \operatorname{span}\{1, x_1, x_2, x_3, x_1x_2\},$$
$$\mathbf{V}(x) = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_1x_2 \end{bmatrix} \otimes \mathbf{I}_s.$$

The missing quadratic monomials are obtained from the selected linear ones, and $x_1x_2x_3 = x_3(x_1x_2)$. To make the hash choices explicit, first take $s = 1$. We use four hash slots, one for each nonconstant selected monomial, and assign them to input labels as follows:

$$H_{j,b} = H(A^b_{i_j}; \tau_j), \qquad (i_1, i_2, i_3, i_4) = (1, 2, 3, 2), \quad b \in \{0, 1\}.$$

Here τ_j contains the gate identifier and the slot index. In particular, slots two and four hash the same input label in different domains. Choose the constant as the parent of each singleton, and $\{1\}$ as the parent of $\{1, 2\}$. In the hash order $(H_{1,0}, H_{1,1}, \ldots, H_{4,0}, H_{4,1})$, the row of coefficients is

$$\mathbf{M}(x) = \begin{bmatrix} 1 + x_1 & x_1 & 1 + x_2 & x_2 & 1 + x_3 & x_3 & x_1(1 + x_2) & x_1x_2 \end{bmatrix}.$$

Each column lies in colspace($\mathbf{V}$) after stacking over all x, and a hash contributes only when its label is available.

At the actual input colors $p = (p_1, p_2, p_3)$, the evaluator can compute $H_{j,p_{i_j}}$; the other value in each pair is unknown. Their coefficient row, still viewed as a function of x, is

$$\mathbf{M}_p(x) = \begin{bmatrix} x_1 + p_1 & x_2 + p_2 & x_3 + p_3 & x_1x_2 + p_2x_1 \end{bmatrix}.$$

Let $\mathbf{M}_p$ be the matrix obtained by stacking these rows over all eight values of x. Reading off coordinates in the ordered basis $(1, x_1, x_2, x_3, x_1x_2)$ gives

$$\mathbf{V}^{-1}\mathbf{M}_p = \left[\begin{array}{cccc|c} p_1 & p_2 & p_3 & 0 & \\ \hline 1 & 0 & 0 & p_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right].$$

S2 removes the first row, which corresponds to the output label. The remaining 4×4 matrix is upper triangular with ones on the diagonal, and hence has rank four for every p. Thus the four unknown hashes independently mask all four transmitted table entries. For s slices, include the slice index in each hash domain and tensor the displayed coefficient matrices with $\mathbf{I}_s$; the same argument gives rank $4s$. This example illustrates the general mechanism; its cost 4λ before control information is larger than that of our dedicated three-input construction.

F.3 Repairing an asymmetric cover

An upward radius-one asymmetric cover $\mathcal{C}_t \subseteq 2^{[t]}$ satisfies

$$J \in \mathcal{C}_t \quad \text{or} \quad J \setminus \{i\} \in \mathcal{C}_t \text{ for some } i \in J \qquad (J \subseteq [t]). \tag{34}$$

In particular, $\varnothing \in \mathcal{C}_t$. For $k \geq 1$, write subsets of $[t] \sqcup [k]$ as pairs (J, L). Begin with $\mathcal{F}_0 = \mathcal{C}_t \times 2^{[k]}$. Whenever a nonempty center $C \in \mathcal{C}_t$ has no parent in $\mathcal{C}_t$, choose any $i(C) \in C$ and add $(C \setminus \{i(C)\}, \varnothing)$. Finally, add any missing singletons, and call the resulting family $\mathcal{F}_{t,k}$.

To check (17), consider an arbitrary nonempty (J, L), whether or not it was selected. If $J \notin \mathcal{C}_t$, condition (34) supplies an old-coordinate parent $(J \setminus \{i\}, L) \in \mathcal{F}_0$. If $J \in \mathcal{C}_t$ and $L \neq \varnothing$, delete a new coordinate to obtain a parent in $\mathcal{F}_0$. Finally, if $J \in \mathcal{C}_t$ and $L = \varnothing$, then $J \neq \varnothing$: either an old-coordinate parent was already in the cover, or the repair step added one. These cases cover all sets, including any newly added repairs, so further rounds of repair are unnecessary.

The product uses $2^k|\mathcal{C}_t|$ sets. There are at most $|\mathcal{C}_t| - 1$ nonempty centers to repair and at most $t + k$ singletons to add. In particular,

$$|\mathcal{F}_{t,k}| \leq (2^k + 1)|\mathcal{C}_t| + t + k + 1. \tag{35}$$

F.4 The uniform constant

Let $K^+(t, 1)$ be the minimum size of a cover in (34). Complementation identifies it with the downward-cover parameter of Cooper, Ellis, and Kahng [8]. Their Propositions 6 and 7 imply, for every power of two t,

$$K^+(t, 1) \leq \beta\, 2^t/t, \qquad \beta = 4\big(\log(\beta/2) + 1\big), \tag{36}$$

where β is the larger root and log is natural. Indeed, the radius-one constant in their Proposition 6 can be chosen as 2, since

$$\sum_{j=0}^{t} \binom{t}{j} \frac{1}{t-j+1} = \frac{2^{t+1}-1}{t+1} < 2\frac{2^t}{t}.$$

The admissibility inequalities for their Proposition 7 then become $\beta \geq 2$ and $\beta \geq 4(\log(\beta/2)+1)$, which give (36). Numerically, $\beta = 10.713387960\ldots < 10.714$.

For $2 \leq n \leq 8$, select the whole family $2^{[n]}$. It satisfies the parent condition and $2^n - 1 \leq 22 \cdot 2^n/n$. For $n \geq 9$, let t be the largest power of two strictly below n, and put $k = n - t$. Then $t \geq 8$ and $1 \leq k \leq t$, including when n itself is a power of two. Combining (35) and (36), with $b_n = |\mathcal{F}_{t,k}|$, gives

$$b_n - 1 \leq \beta(2^k + 1)2^t/t + n.$$

After division by $2^n/n$, the right-hand side is

$$\beta(1 + 2^{-k})(1 + k/t) + n^2/2^n. \tag{37}$$

To bound this expression uniformly, let $g_t(k) = (1 + 2^{-k})(1 + k/t)$. Its successive difference is

$$g_t(k+1) - g_t(k) = 1/t - 2^{-(k+1)}\big(1 + (k-1)/t\big).$$

The subtracted term decreases with k, so the differences increase; hence the maximum on $1 \leq k \leq t$ occurs at an endpoint. For $t \geq 8$,

$$g_t(1) \leq 27/16, \qquad g_t(t) = 2(1 + 2^{-t}) \leq 257/128.$$

Also $n^2/2^n$ decreases for $n \geq 3$, and is therefore at most $81/512$ when $n \geq 9$. Thus (37) is less than

$$10.714 \cdot 257/128 + 81/512 = 21.66990625 < 22.$$

This proves $b_n - 1 \leq 22 \cdot 2^n/n$ for every $n \geq 2$.

F.5 Compressed controls and joint security

We now take $s = 1$, write $N = 2^n$ and $b = |\mathcal{F}|$, and use $\mathbf{P}, \mathcal{N}, J$ from Section 6.1, with the full message space $\mathcal{D} = \mathcal{Q}$. Here $\mathbf{P}$ has $u = N - b$ rows. Concretely, it can extract the algebraic-normal-form coefficients indexed by sets outside $\mathcal{F}$: these coefficients vanish exactly on colspace($\mathbf{V}$). Thus $\ker \mathbf{P} = \text{colspace}(\mathbf{V})$, and $\mathcal{N}$ maps the n control coefficient vectors to their u-bit syndrome. When $b = N$, the syndrome is empty and the control cost is zero. The right inverse J is fixed by the public matrices, independently of the gate and the main garbled table.

The control encoder. For each nonempty $I \in \mathcal{F}$ and $c \in \{0,1\}$, let $a_{I,c} \in \mathbb{F}_2^b$ be the coefficient vector of the supported column $h_{I,1,c}$ defined above. Thus $\mathbf{V}(x)a_{I,c} = 0$ whenever $x_{i(I)} \neq c$. For each control index $j \in [n]$, obtain a separate one-bit pad $\rho_{j,I,c}$ by hashing input label $A^c_{i(I)}$. The tweak identifies the gate, the control purpose, j, and I. Control, main-table, and output-decoding queries use disjoint domains. Put

$$h_j = \sum_{I \in \mathcal{F}\setminus\{\varnothing\}} \sum_{c\in\{0,1\}} a_{I,c}\rho_{j,I,c}, \qquad h = (h_1, \ldots, h_n).$$

Send $\mathbf{z} = \gamma(\tilde{g}) + \mathcal{N}(h)$ and use the controls $F = h + J\mathbf{z}$. The evaluator obtains $\mathbf{V}(p)h_j$ from its known pads, since every unknown pad has a coefficient vanishing at p. It can therefore recover the local controls from $\mathbf{z}$. The identity $\mathcal{N}(F) = \gamma(\tilde{g})$ proves correctness.

Uniformity of the control message. At a fixed color p, the $b-1$ unknown supported columns form a basis of $\ker \mathbf{V}(p)$: their projections onto the nonconstant coordinates are the invertible triangular matrix used to prove S2. Independent copies for the n control indices therefore span $\mathcal{A}_p = \{(Q_1, \ldots, Q_n) : \mathbf{V}(p)Q_j = 0 \text{ for all } j\}$. Moreover, $\mathcal{N}(\mathcal{A}_p) = \mathbb{F}_2^u$. To see this directly, start with any preimage $F = (Q_1, \ldots, Q_n)$, which exists by C, and replace Q_j by $Q_j + (\mathbf{V}(p)Q_j)e_\varnothing$, where $e_\varnothing$ is the coefficient vector of the constant column. The local values become zero, while the syndrome is unchanged: $\mathbf{P}\mathbf{X}_j\mathbf{V}e_\varnothing = 0$ because every singleton monomial $m_{\{j\}}$ is selected. Hence $\mathbf{z}$ is uniform conditioned on all known control pads. The complete supported family also spans $\mathbb{F}_2^b$: it contains every nonconstant selected monomial, and the two columns for any singleton sum to the constant column. Consequently h is uniform, and $F = (I + J\mathcal{N})h + J\gamma(\tilde{g})$ has the original uniform distribution on correct control tuples.

Joint security with the main table. Use independently domain-separated λ-bit main-table hashes. Condition on the input labels, offset, all control pads, and known main-table hashes. S2 makes the map from the $b-1$ unknown hashes to $\mathbf{G}$ invertible. Thus $\mathbf{G}$ is uniform under this conditioning, and $(\mathbf{z}, \mathbf{G})$ is jointly uniform given the known pads. The evaluator obtains its local controls from $\mathbf{z}$ and its known control pads.

Replace Hybrid 1 of Appendix A as follows. Sample $\mathbf{z}$ and $\mathbf{G}$ uniformly, sample the unknown control pads uniformly subject to $\mathcal{N}(h) = \gamma(\tilde{g}) + \mathbf{z}$, then complete the unknown main-table hashes using S2. Surjectivity makes this a reordering of the real sampling procedure. The remaining hybrids defer Δ and omit the programming, as before. Each programmed input is an active label plus Δ, with a tweak identifying its gate, slot, and copy. The same random-oracle guessing bound therefore applies. Privacy and authenticity follow with the unchanged, separately domain-separated output decoder. This proves joint security for the unsliced full syndrome encoder.

Communication. The main table has $b-1$ entries of length λ, and the control syndrome contains $N-b$ bits. The exact per-gate count is therefore

$$(b-1)\lambda + N - b = (b-1)(\lambda-1) + N - 1.$$

For $\lambda \geq 1$, the previously proved bound $b - 1 \leq 22N/n$ gives

$$(b-1)\lambda + N - b \leq 22N(\lambda-1)/n + N - 1 \leq 22N\lambda/n + N.$$

No slice padding is needed. All queries use input labels and public tweaks, so they remain nonadaptive. This proves Theorem 4.

G Control Encoding Details

We give the independent-pad arguments behind Section 6.1. One encoder covers both the full syndrome space and the smaller three-input message space. We first state its conditions, then instantiate them and recall the separate-column zero-row variant. The final subsection proves the supported-mask spanning property used by the full encoder.

G.1 Choosing the message space

Use the notation $\mathcal{N}, \gamma, \mathcal{A}, \mathcal{Q}$ from Section 6.1. Put $\mathcal{T} = \mathrm{span}\{\gamma(g) : g : \{0,1\}^n \to \mathbb{F}_2\}$ and $\mathcal{K} = \ker \mathcal{N}$. All spaces and maps below are public and fixed independently of the gate.

For each pad class α, choose a subspace $\mathcal{L}_\alpha \subseteq \mathcal{A}$ whose local control matrices vanish at every color where those pads are unavailable. Independent pad bits multiply a basis of each $\mathcal{L}_\alpha$; distinct bits use separate domains. Let $\mathcal{L}$ be their total span and $\mathcal{L}_p$ the span of classes unavailable at p. Choose $\mathcal{T} \subseteq \mathcal{D} \subseteq \mathcal{Q}$ and a linear correction map $J : \mathcal{D} \to \mathcal{A}$ satisfying

$$\mathcal{N}(\mathcal{L}_\alpha) \subseteq \mathcal{D}, \qquad \mathcal{N}(\mathcal{L}_p) = \mathcal{D} \quad \text{for every } p,$$
$$\mathcal{N}J = I_{\mathcal{D}}, \qquad (I + J\mathcal{N})\mathcal{L} = \mathcal{K}.$$

No requirement that $J(\mathcal{D}) \subseteq \mathcal{L}$ is needed.

The summed mask h is uniform on $\mathcal{L}$. The encoder (20) sends $\mathbf{z} = \gamma + \mathcal{N}(h)$ and sets $F = h + J\mathbf{z}$. Correctness follows from $\mathcal{N}(F) = \gamma$. Conditioned on the known pads at p, the unknown mask is uniform on $\mathcal{L}_p$, so its syndrome is uniform on $\mathcal{D}$. Thus $\mathbf{z}$ is uniform on $\mathcal{D}$ independently of the gate. Unknown masks vanish locally, making the local controls a function of $\mathbf{z}$ and the known pads; their joint view is also gate-independent. Finally,

$$F = J\gamma + (I + J\mathcal{N})h$$

is uniform on the complete correctness fiber $J\gamma + \mathcal{K}$. Sending coordinates in a public basis of $\mathcal{D}$ costs $\dim \mathcal{D}$ bits. These statements concern independent pads, with the security scope stated in Section 6.1.

G.2 Correctness and masking of the syndrome encoder

The full syndrome encoder is the special case $\mathcal{D} = \mathcal{Q}$. Condition C makes $\mathcal{N}$ surjective, so choose any public right inverse $J : \mathcal{Q} \to \mathcal{A}$. Let t be the number of columns of $\mathbf{M}$. For each input i, let $\mathbf{H}_i^{\mathrm{ctrl}} \in \mathbb{F}_2^{t \times s}$ contain fresh control-hash bits, with rows corresponding to the hash inputs of $\mathbf{M}$. Separate domains distinguish the gate, hash slot, control input index, source column, and main-table hashes. Take

$$h_i = (\mathbf{V}^{-1}\mathbf{M})\mathbf{H}_i^{\mathrm{ctrl}}, \qquad h = (h_1, \ldots, h_n),$$

where $\mathbf{V}^{-1}$ is a left inverse. The support of each column of $\mathbf{M}$ ensures that its local contribution vanishes whenever the corresponding hash is unavailable.

S1 ensures that masks invisible at p can realize every syndrome. For a fixed p, let $\mathcal{A}_p$ contain the tuples F satisfying $\mathbf{V}(p)\mathbf{Q}_i = 0$ for all i. Given any syndrome, C supplies a preimage F, and S1 supplies a homogeneous solution with the same local controls at p. Adding them gives a preimage in $\mathcal{A}_p$. Hence $\mathcal{N}(\mathcal{A}_p) = \mathcal{Q}$. By S2, the unknown supported columns at p span $\ker \mathbf{V}(p)$ in coefficient coordinates. Their independent copies therefore give $\mathcal{L}_p = \mathcal{A}_p$ and $\mathcal{N}(\mathcal{L}_p) = \mathcal{Q}$. S1 and S2 also ensure that the complete supported family spans all coefficient directions, so $\mathcal{L} = \mathcal{A}$; a short proof is given in Appendix G.4. Since $(I + J\mathcal{N})\mathcal{A} = \ker \mathcal{N}$, all conditions of Appendix G.1 hold. This instance transmits $\dim \mathcal{Q} = us = s(s2^n - d)$ bits.

The separate-column zero-row encoding. The equality $\mathcal{N}(\mathcal{A}_0) = \mathcal{Q}$ allows a public right inverse J_0 whose image lies in $\mathcal{A}_0$. Put $\mathcal{K}_0 = \ker \mathcal{N} \cap \mathcal{A}_0$ and sample k_0 uniformly in $\mathcal{K}_0$, independently of the hash masks. The zero-row encoding transmits the coefficients of

$$Z_0 = J_0\big(\gamma(\tilde{g}) + \mathcal{N}(h)\big) + k_0, \qquad F = h + Z_0.$$

Since $Z_0 \in \mathcal{A}_0$, the local controls at the origin agree with the local masks. Each control column's correction lies in the r-dimensional subspace of colspace($\mathbf{V}$) vanishing there. There are ns such columns, so this version transmits nsr bits. Conditioned on the known pads at any p, the syndrome is uniform in $\mathcal{Q}$ and the independent k_0 fills the kernel of $\mathcal{N}$ on $\mathcal{A}_0$. Thus Z_0 is uniform on $\mathcal{A}_0$ independently of the gate. Unconditionally, $(I + J_0\mathcal{N})h$ is uniform on $\ker \mathcal{N}$, so adding k_0 preserves the uniform correct-control sampler. This is an independent-pad argument, under the same assumptions as the full syndrome encoding. For $(n, s, d, r) = (3, 3, 10, 7)$, the count is $3 \cdot 3 \cdot 7 = 63$ bits.

G.3 The restricted three-input encoder

Fix the three-input instance in (13). Its parameters are $s = 3$, $d = 10$, $r = 7$, and $u = 14$, so the full syndrome has 42 bits. We now retain the finite linear-algebra calculation underlying the further reduction to 35 bits.

Let

$$\mathcal{T} = \operatorname{span}\{\gamma(g) : g : \{0,1\}^3 \to \mathbb{F}_2\} \subseteq \mathcal{Q}.$$

Exact elimination gives $\dim \mathcal{T} = 7$: the only gate tables with zero joint syndrome are the two constant tables. This does not, however, make a seven-bit encoding possible by the preceding method. The masks must still obey the evaluator's hash-availability pattern.

To express that restriction, for a nonzero binary linear form ℓ and $b \in \{0,1\}$, let $\mathcal{C}_{\ell,b}$ be the subspace of control tuples whose local control matrices vanish whenever $\ell(x) \neq b$, and put $\mathcal{B}_{\ell,b} = \mathcal{N}(\mathcal{C}_{\ell,b})$. These are the syndrome directions a pad supported on the halfspace $\ell = b$ can mask. At every color vector p, the seven

spaces $\mathcal{B}_{\ell,1-\ell(p)}$ form a direct-sum decomposition of $\mathcal{Q}$ for this construction. Let $P_{p,\ell}$ denote the projection onto the corresponding component.

Suppose the public message is restricted to a fixed linear subspace $\mathcal{D} \subseteq \mathcal{Q}$ containing $\mathcal{T}$. Each gate target must be hidden by the unknown masks at every p. Uniqueness of the direct-sum decomposition then requires $P_{p,\ell}(\mathcal{T}) \subseteq \mathcal{D}$ for every p, ℓ. Consequently the smallest possible message space in this format is

$$\mathcal{D}_0 = \operatorname{span}_{p,\ell} P_{p,\ell}(\mathcal{T}). \tag{38}$$

This gives a finite linear-algebra calculation, rather than a search over all possible encodings. For the fixed three-input matrices, $\dim \mathcal{D}_0 = 35$. The seven summands of the hidden syndrome decomposition intersect $\mathcal{D}_0$ in dimensions $6, 5, 1, 6, 5, 6, 6$, respectively, when the forms are ordered as $x_1, x_2, x_1+x_2, x_3, x_1+x_3, x_2+x_3, x_1+x_2+x_3$. These dimensions sum to 35, and the certificate checks their independence at each of the eight color vectors. For example, the mask component associated with $x_1 + x_2$ needs only one syndrome coordinate rather than six. The supported masks therefore span all 35 transmitted directions. Thus $\mathbf{z}$ in (20) can be transmitted using 35 bits.

We also check that this reduction preserves the control sampler. Let $\mathcal{L}_0$ be the span of the supported control tuples whose syndromes lie in $\mathcal{D}_0$. The finite certificate gives

$$\dim \mathcal{L}_0 = 83, \qquad \dim \ker \mathcal{N} = 48, \qquad (I + J\mathcal{N})\mathcal{L}_0 = \ker \mathcal{N}.$$

Here J is a public right inverse on $\mathcal{D}_0$ with image in $\mathcal{L}_0$. The local evaluation of the homogeneous controls has rank 27 at every point, as required for the three 3×3 control matrices. These identities, the mask supports, and all eight hidden-syndrome ranks are verified using the literal finite matrices. The value 35 is minimal within this fixed linear decoder format, with independent scalar pads multiplying public halfspace-supported vectors. It is not a lower bound for arbitrary control encodings.

G.4 Spanning of the supported masks

We prove the spanning claim used in Section 6.1. Write $S = \mathrm{colspace}(\mathbf{V})$ and $W_p = \{f \in S : f(p) = 0\}$. Since S contains all constant tables, $\dim W_p = d - s = r$. The unknown hash columns at p lie in W_p; by S2, their images modulo constants have rank r. They therefore span W_p.

Fix distinct colors $p, q \in \{0,1\}^n$. We show that $W_p + W_q = S$. Suppose evaluation of W_p at q were not onto. There would be a nonzero linear functional ω on $\mathbb{F}_2^s$ vanishing on all these values. For any $f \in S$, the table $x \mapsto f(x) + f(p)$ belongs to W_p, giving $\omega f(q) = \omega f(p)$. Now take an arbitrary homogeneous correctness relation, using $x \in \{0,1\}^n$ as the formal variable:

$$f_0(x) + \sum_{i=1}^{n} x_i f_i(x) = 0, \qquad f_i \in S.$$

Apply ω at p and q, and add the equations. The preceding identity yields

$$\sum_{i=1}^{n} (q_i + p_i)\omega f_i(p) = 0.$$

Because $p \neq q$ and $\omega \neq 0$, this contradicts S1, which permits every tuple of local values $(f_1(p), \ldots, f_n(p))$. Thus evaluation of W_p at q is onto. Since W_q is the kernel of evaluation at q on S, this proves $W_p + W_q = S$.

Thus the complete supported family spans S. Independent copies for each input index and source slice make h uniform on $\mathcal{A}$. Conditioned on the known pads at p, the unknown copies span $\mathcal{A}_p$, as required by the syndrome argument.

H Local Rewriting Details for the Circuit Evaluation

This appendix records the local synthesis procedure used in Section 6.2. The candidate pool also includes verified networks from the published three-input synthesis flow cited there; all candidates use the same component costs and compete with the selected binary fallback.

For the local replacements, we start from a binary circuit and simplify AND regions and small cuts while retaining the resulting binary candidate. We then enumerate single-output regions with at most three boundary wires, allowing XOR nodes inside the region. We evaluate each region's truth table exhaustively and compare its minimum two-input AND cost with C_3. We accept a replacement only if it reduces communication and every removed internal wire has no consumer outside the region. Several deterministic greedy orders are tried. A complementary pass reassociates unshared positive AND regions and covers them with binary and ternary AND gates. Cut enumeration is limited to 20 seconds per round; an unfinished enumeration is discarded and replaced by complete direct AND-pair enumeration.

These choices make the procedure straightforward: it recognizes small opportunities already present in a circuit and keeps a replacement only when its

immediate cost is lower. It does not jointly optimize the circuit representation and the placement of three-input gates. In particular, shared intermediate values can prevent an otherwise attractive replacement, and a greedy choice can exclude a better combination of later replacements.