Rerandomizable Garbling for Arithmetic Circuits

Anasuya Acharya* Pierre Meyer† Claudio Orlandi‡ Lawrence Roy§

Abstract

Rerandomizable garbling is a powerful extension of Yao's garbled circuits that allows a garbled circuit to be refreshed without changing its functionality, while preserving privacy as long as at least one rerandomization step is honest. Existing constructions, however, are limited to Boolean circuits and incur significant communication overhead. In this work, we initiate the study of rerandomizable garbling for arithmetic circuits. We present the first rerandomizable arithmetic garbling scheme, based on techniques inspired by recent advances in garbling from homomorphic secret sharing. Our construction departs fundamentally from previous approaches and achieves communication complexity $O(|\mathbf{C}| \log B)$ for circuits $\mathbf{C}$ with B-bounded computation. Concretely, our scheme requires 17 group elements per multiplication gate, while addition gates are free. Our scheme can be instantiated under standard subgroup-indistinguishability assumptions, including the Decisional Composite Residuosity (DCR) assumption and assumptions over class groups.

1 Introduction

Garbling schemes are a central cryptographic primitive, first introduced by Yao as part of his two-party computation protocol [Yao86]. They were later formalized and studied as cryptographic objects in their own right [BHR12]. At a high level, a garbling scheme allows one to "encrypt a computation": a circuit is transformed into a garbled circuit, which can be evaluated on garbled inputs to produce a garbled output that can subsequently be decoded. Garbled circuits have found numerous applications and are by now a standard cryptographic primitive.

An interesting and powerful variant of garbling schemes is *rerandomizable* garbling. Introduced by Gentry et al. [GHV10], rerandomizable garbling allows a party to transform an existing garbled circuit into a fresh-looking one that includes its own randomness without changing the underlying functionality. Privacy is preserved as long as at least one rerandomization step is honest. Rerandomizable garbling enables multiparty garbling in specialized communication models where parties communicate sequentially. Moreover, the size of the resulting garbled circuit does not grow with the number of parties. Crucially, rerandomization can be performed by any party holding only the garbled circuit, and requires no knowledge of the original garbler's secrets or randomness. This is what makes rerandomizable garbling more than simply re-garbling from scratch, and it enables powerful applications as described in Section 1.1 below.

In most of these applications, rerandomizable garbling for Boolean circuits can be replaced by rerandomizable garbling for arithmetic circuits, simply by replacing the corresponding variant of OT with OLE. However, despite recent advances [HvdHJ25], existing constructions of rerandomizable garbling are limited to Boolean circuits and incur significant communication overhead.

The arithmetic setting is well motivated on its own. Many privacy-preserving machine learning and secure aggregation tasks are naturally expressed as arithmetic computations, while

*Aalborg University, Denmark. anasuyaa@es.aau.dk
†CISPA Helmholtz Centre for Information Security, Germany. pierre.meyer@cispa.de
‡Aarhus University, Denmark. orlandi@cs.au.dk
§IBM Research Zürich, Switzerland. ldr709@gmail.com

implementing them using Boolean circuits can introduce significant overhead. More generally, arithmetic circuits are a natural representation for a wide range of secure computation tasks.

Of course, any arithmetic computation can in principle be represented as a Boolean circuit. The relevant question is therefore not merely whether rerandomizable garbling for arithmetic computations is possible, but whether it can be achieved natively and efficiently in the arithmetic setting. Concretely, we ask:

Can we construct rerandomizable garbling for arithmetic circuits, and at what communication cost?

To answer this question, we draw inspiration from a seemingly orthogonal, but closely related, line of work on advanced garbling techniques based on homomorphic secret sharing and distributed discrete logarithms [BGI16, BGI+18, OSY21, RS21]. This line of work has led to great improvements in the communication complexity of garbling, including one bit per ciphertext [LWYY25], sublinear-size garbling [ILL24, ILL25], and constant-rate arithmetic garbling [MORS24, MORS25b, MORS25a].

Inspired by these techniques, we construct *the first rerandomizable garbling scheme for arithmetic circuits*. Our construction achieves communication complexity $O(|\mathbf{C}| \log B)$ for circuits $\mathbf{C}$ with B-bounded computation. Concretely, the garbled circuit uses 17 group elements per multiplication gate, while addition gates are free. For sufficiently large arithmetic circuits, this gives constant-rate garbling in the sense of [MORS24] (albeit with larger constants).

The most communication-efficient prior rerandomizable garbling scheme is [HvdHJ25]. Compared to our approach, their construction incurs a quadratic blowup in the security parameter, in addition to the overhead required to compile arithmetic circuits into Boolean ones.

Our framework can be instantiated under standard subgroup-indistinguishability assumptions, including the Decisional Composite Residuosity (DCR) assumption and assumptions arising from class groups as in [CL15, ADOS22].

1.1 Applications of Rerandomizable Garbling

Rerandomizable garbling allows for distributed garbling in the presence of all-but-one corrupt garblers in settings with communication constraints. It has many applications, including:

i-hop homomorphic encryption [GHV10]. This is a setting involving multiple parties with secret functions wanting to securely compute the output of the sequential *composition* of all these functions over a private input, without revealing their individual functions. Rerandomizable garbling can be used to securely compose the garbled circuit of a function over prior garbled functions in such a way that even the collusion of *all* other (past and future) function holders does not reveal the function underlying this garbling.[1]

Secure computation with reverse firewalls [MS15]. Reverse firewalls are a cryptographic object that interfaces between an end user and the network, protecting the user's data when they are executing possibly compromised software that communicates with an adversary in the network. Rerandomizable garbling, along with mechanisms that ensure malicious security, are used to instantiate these firewalls to ensure that honest functionality in the network protocol is maintained on the user's part regardless of whether its local software is compromised.

MPC in the SCALES framework [AHKP24, AHKP25]. Rerandomizable garbling is a central tool for enabling outsourced secure computation for any functionality with ephemeral servers in the SCALES model. Protocols in the SCALES model use this to perform distributed garbling among servers that can each post only one message on-the-fly, and assuming all-but-one corruption.

[1] As is standard, function privacy is obtained by garbling a universal circuit, since our scheme (like most garbling schemes supporting free linear gates) reveals the circuit structure.

Outsourced re-garbling [AHKP22]. For applications wherein a secret function needs to be securely evaluated many times over different inputs, rerandomizable garbling allows the function holder to garble its circuit *only once* and outsource the task of replicating it for different evaluations to a more powerful server that can do so via rerandomizing. This essentially allows a relaxed form of reusability for garbling, relying on computational assumptions that are weaker than the assumptions known to imply reusable garbling (iO, LWE).

A common feature of all these settings is that the party performing the rerandomization is *not* the original garbler, and does not hold its secrets: it may be a subsequent function holder, a firewall, an ephemeral server, or an untrusted worker. Those are all settings in which re-garbling from scratch is not an option.

Black-box zero-knowledge proofs of correct garbling and rerandomizing [AHKP24]. A garbler (resp. rerandomizer) can non-interactively prove in zero-knowledge that there exists valid randomness for which the presented garbled circuit is a correct garbling of the given function (resp. correct rerandomizing of the given garbled circuit). This allows round-optimal *maliciously secure* function evaluation in 2PC, in the plain and CRS models. The malicious overhead in such protocols is that of a multiplicative soundness parameter over the original garbled circuit. All existing rerandomizable garbling schemes for Boolean circuits allowed for this due to an implicit fact in the definition as a result of which all the garbling randomness used would form a finite group with 'rerandomizing' being a group action. Extending this to be applicable for garbling arithmetic circuits makes this group infinite and therefore we make this structure explicit in our restatement of the definition of a rerandomizable garbling scheme.

1.2 Related Work

Rerandomizable Garbling Schemes. [GHV10] introduce rerandomizable garbling (as part of rerandomizable secure function evaluation) and propose a construction, which they use for multi-hop homomorphic encryption. [MS15] construct a variant of rerandomizable garbling from El Gamal for layered Boolean circuits, also incorporating security against a malicious garbler using authentication tags; they apply it to build reverse firewalls. [AHKP22, AHKP24] formalize the notion of rerandomizable garbling and fix a bug in the construction and proof of [GHV10]. They propose multiple additional applications and obtain garbling size $O(|\mathbf{C}|\kappa^3)$ assuming DDH for Boolean circuits; in particular, they treat (semi-honest) rerandomizable garbling as a cryptographic object and show that black-box constructions imply zero-knowledge proofs of knowledge of correct garbling and rerandomization. [HvdHJ25] improve the garbling size to $O(|\mathbf{C}|\kappa^2)$ assuming pairings, again for Boolean circuits. Finally, [Kil88, AHKP25] build rerandomizable garbling from permutation branching programs rather than circuits: [AHKP25] formalize information-theoretic rerandomizable garbling (a construction implied in [Kil88]), where the construction has input label size $O(|f|)$, and then use it to obtain RGS with garbling size $O(|f|\kappa)$ and input label size $O(\kappa)$ under DDH (based on El Gamal). None of these constructions achieve constant rate or support free-XOR.

Arithmetic Garbling. [AIK11] gave the first direct method to garble arithmetic computations (i.e., without emulating them as Boolean circuits) under bounded integer computation. Subsequent works improved efficiency and rate, culminating in constant-rate arithmetic garbling under assumptions such as DCR [BLLL23, HY24, MORS24]. None of these constructions appear to directly extend to the rerandomizable setting. In terms of concrete communication overhead, our construction is strictly worse than [MORS24] (which achieves rate 1 and free additions), and achieves similar communication complexity to [BLLL23] (our construction uses 17 group elements per multiplication gate, while theirs uses between 12 and 18 depending on the concrete instantiation; both constructions may admit further optimizations), while additionally supporting free additions.

1.3 Technical Overview

Existing rerandomizable garbling schemes are inherently Boolean. At a high level, they rely on "symmetric-key" wire labels that are encrypted under a circular-secure encryption scheme (the BHHO [BHHO08] scheme and derivatives [HvdHJ25]). Rerandomization is achieved by permuting input labels and correspondingly permuting ciphertext components. In such schemes, permuting the group elements inside a BHHO ciphertext effectively permutes the underlying encrypted messages. Combined with an appropriate restriction on the symmetric keys (i.e., enforcing an equal number of 0s and 1s in labels), this yields a way to rerandomize input labels and the corresponding ciphertexts while preserving functionality.

This approach, however, does not naturally extend to arithmetic circuits. Our construction takes a fundamentally different approach. Instead of permuting discrete labels, we interpret rerandomization as an affine transformation of algebraic encodings. This perspective allows rerandomization to act linearly on both labels and ciphertexts, making it compatible with arithmetic computation, and arguably conceptually simpler than previous works.

We now present the main ideas behind our scheme. At a high level, our rerandomizable arithmetic garbling construction is built around a novel algebraic gadget that supports affine transformations both of the function being garbled and of the underlying randomness. In this technical overview we sacrifice precision for clarity and defer the formal treatment to later sections. (In the full construction, all objects are instantiated over the integers together with structured group embeddings, and careful parameter bounds are required; here we suppress these details to focus on the core ideas.)

Labels. Each wire carries a *label* encoding a value x. The encoding consists of two components. First, we mask the value:

$$\bar{x} = x + m,$$

where m is a wire-dependent mask. Next, we attach an affine authentication tag:

$$X = \Delta\bar{x} + K,$$

where Δ is a *global key* shared across all wires in the circuit, and K is a wire-specific key.[2] Thus a label has the form

$$[\bar{x}, X].$$

Free Addition. The linear structure immediately implies that additions are free. Given encodings $[\bar{x}_1, X_1]$ and $[\bar{x}_2, X_2]$, we obtain an encoding of $x_3 = x_1 + x_2$ by componentwise addition:

$$\bar{x}_3 = \bar{x}_1 + \bar{x}_2, \qquad X_3 = X_1 + X_2.$$

It is immediate that the resulting pair is again a valid label:

$$X_3 = \Delta\bar{x}_3 + (K_1 + K_2),$$

Multiplication via Bilinear Forms. As usual in garbling schemes, the difficult part is handling multiplication gates. Our construction allows, in fact, to garble more expressive functions, namely any bilinear function of the form:

$$f(\bar{x}_1, \bar{x}_2) = a\,\bar{x}_1\bar{x}_2 + b\,\bar{x}_1 + c\,\bar{x}_2 + d.$$

Such a function can always be represented in matrix form as

$$f(\bar{x}_1, \bar{x}_2) = \begin{bmatrix} 1 & \bar{x}_1 \end{bmatrix} F \begin{bmatrix} 1 \\ \bar{x}_2 \end{bmatrix}, \text{ where } F = \begin{bmatrix} d & c \\ b & a \end{bmatrix}.$$

[2] Readers familiar with "free-XOR" [KS08] style garbling will recognise Δ as playing the role of the global offset, and K as the wire key. The masked value $\bar{x}$ plays a role similar to the colored bit in those schemes.

Note in particular that such forms are sufficient to evaluate a gate that unmasks its inputs, multiplies them, and masks the output e.g.,

$$f(\bar{x}_1, \bar{x}_2) = m_3 + (\bar{x}_1 - m_1)(\bar{x}_2 - m_2)$$

This property is crucial in our construction, as it allows to perform multiplications of masked inputs, returning a masked output, thus preserving the invariant.

Cryptographic Tools. Before moving on, we need to introduce a cryptographic tool that will be used for garbling: we introduce a new framework, which we call *subgroup indistinguishability with efficient discrete logarithms (SIEDL)*, that generalizes both the Damgård–Jurik cryptosystem [DJ01] and cryptosystems based on class groups [CL15]. At a high level, SIEDL consists of a group $\mathbb{G}$ equipped with two distinguished subgroups $\mathbb{F}$ and $\mathbb{H}$. The key computational assumption is *subgroup indistinguishability*, namely random elements from $\mathbb{G}$ are computationally indistinguishable from random elements from the subgroup $\mathbb{H}$.

The subgroup $\mathbb{F}$ has known prime order q, and the discrete logarithm problem in $\mathbb{F}$ is easy. We denote by f a generator of $\mathbb{F}$, and by g a random element sampled from $\mathbb{G}$. We use additive notation for all groups, which simplifies the algebraic presentation and aligns naturally with our affine view of garbling.

Encrypting a Bilinear Function. We are now ready to present our garbling scheme for bilinear functions. Let $F \in \mathbb{Z}^{2\times 2}$ define a function

$$f(\bar{x}_1, \bar{x}_2) = \begin{bmatrix} 1 & \bar{x}_1 \end{bmatrix} F \begin{bmatrix} 1 \\ \bar{x}_2 \end{bmatrix}.$$

We construct an encryption algorithm that turns F into a ciphertext $C \in \mathbb{G}^{3\times 3}$ satisfying the decryption correctness property

$$f(\bar{x}_1, \bar{x}_2) = \mathsf{DLog}_{\mathsf{f}} \left(\begin{bmatrix} 1 & \bar{x}_1 & X_1 \end{bmatrix} C \begin{bmatrix} 1 \\ \bar{x}_2 \\ X_2 \end{bmatrix} \right),$$

where $X_i = \Delta \bar{x}_i + K_i$ are the affine labels and $\mathsf{DLog}_{\mathsf{f}}(\cdot)$ denotes discrete logarithm in the easy subgroup $\mathbb{F}$ with generator f.

Such a ciphertext can be constructed as

$$C = \begin{bmatrix} F & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \mathsf{f} \;+\; \mathsf{pk}_1 \mathbf{u}_1^\top \;+\; \mathbf{u}_2 \mathsf{pk}_2^\top,$$

where $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{Z}^3$ are random scalar vectors and the public-keys are vectors of group elements.[3] The ciphertext must satisfy two properties:

1. (*Privacy.*) It hides the bilinear function F.
2. (*Correctness.*) When evaluating the vector–matrix–vector product, all components outside the easy discrete-log subgroup $\mathbb{F}$ must cancel out.

[3] In the full construction the public keys are instead carried in an additional row and column of the ciphertext, which is therefore an element of $\mathbb{G}^{4\times 4}$.

Correctness. We start arguing that, if this cancellation holds, the resulting group element equals

$$f(\bar{x}_1, \bar{x}_2)\,\mathsf{f},$$

which lies in $\mathbb{F}$ and can therefore be recovered by taking a discrete logarithm. To ensure that the $\mathbb{G}$-components disappear, we require the public keys $\mathsf{pk}_1, \mathsf{pk}_2$ to be orthogonal to the wire labels. If we set

$$\mathsf{pk}_i = \begin{bmatrix} K_i\mathsf{g} \\ \Delta\mathsf{g} \\ -\mathsf{g} \end{bmatrix},$$

then we observe that (since $X_i = \Delta\bar{x}_i + K_i$)

$$\mathsf{pk}_i^\top \begin{bmatrix} 1 \\ \bar{x}_i \\ X_i \end{bmatrix} = K_i\mathsf{g} + \Delta\bar{x}_i\mathsf{g} - X_i\mathsf{g} = 0.$$

Privacy. With regards to the security of the encryption scheme ("C hides F") note that the ciphertext C hides the bilinear form F because each entry of the embedding $F \cdot \mathsf{f}$ is additively masked by randomized multiples of g. Under the SIEDL subgroup indistinguishability assumption, these masking terms are computationally indistinguishable from uniform elements of $\mathbb{G}$, and therefore the resulting distribution of C hides F.

Carrying the Authentication Tag. To preserve the invariant that each wire carries not only a mask $\bar{x}$ but also an affine authentication tag, we must also compute a valid output tag X_3 alongside the output masked value $\bar{x}_3$. Concretely, for an output wire we want again a label of the form

$$\bar{x}_3 = x_3 + m_3, \qquad X_3 = \Delta\bar{x}_3 + K_3,$$

for a fresh wire key K_3 (and the same global key Δ).

A particularly elegant aspect of the construction is that the authentication tag requires no special treatment: since our encryption scheme supports *arbitrary* bilinear functions, and since X_3 can be expressed as a bilinear function of the inputs $\bar{x}_1, \bar{x}_2$, we can simply produce an additional ciphertext that encrypts the corresponding bilinear form for the tag computation. Thus, the tag computation requires no extra technical effort, and is a simple byproduct of the algebraic expressiveness of the framework.

Rerandomizing Labels and Ciphertexts. We are now finally ready to explain how to rerandomize labels and the corresponding ciphertexts. Let the original label vector be

$$\mathbf{v} = \begin{bmatrix} 1 \\ \bar{x} \\ X \end{bmatrix}$$

To rerandomize, we sample fresh wire-shift parameters m', K' together with a fresh shift Δ' of the global key, and define the lower-triangular matrix

$$R' = \begin{bmatrix} 1 & 0 & 0 \\ m' & 1 & 0 \\ (K' + \Delta' m') & \Delta' & 1 \end{bmatrix}.$$

We then set the rerandomized label to be

$$\mathbf{v}' \;=\; R'\,\mathbf{v}.$$

A direct computation gives $\bar{x}' = \bar{x} + m'$ and $X' = X + K' + \Delta'\bar{x}'$, so that $\mathbf{v}'$ has the same syntactic form as $\mathbf{v}$: it is again a valid label, now with respect to the updated global key $\Delta + \Delta'$ and with fresh wire-dependent parameters. Note that R' is built entirely from freshly sampled randomness;[4] in particular, rerandomizing a label requires no knowledge of the previous global key.

Crucially, the ciphertext can be transformed consistently with the label rerandomization. Let $\mathbf{v}_1, \mathbf{v}_2$ be the original input label vectors, and rerandomize them as

$$\mathbf{v}_1' = R_1\,\mathbf{v}_1, \qquad \mathbf{v}_2' = R_2\,\mathbf{v}_2,$$

where R_1, R_2 are of the above form. The wire-specific parameters may differ from wire to wire, but the global-key shift Δ' is sampled once and applied to *every* wire in the circuit: this is what keeps a single global key well defined after rerandomization, and in particular what keeps addition free, since adding two labels under different global keys would not yield a valid label. We then define the rerandomized ciphertext as

$$C' \;=\; (R_1^{-1})^\top\, C\, R_2^{-1}.$$

Correctness is preserved since the randomization matrices cancel with each other:

$$(\mathbf{v}_1')^\top C' \mathbf{v}_2' = (\mathbf{v}_1)^\top R_1^\top (R_1^{-1})^\top C R_2^{-1} R_2 \mathbf{v}_2 = (\mathbf{v}_1)^\top C \mathbf{v}_2.$$

Note that this procedure is entirely public and can be performed by any party holding a garbled circuit using freshly sampled randomness: rerandomization only requires the garbled circuit itself, and needs neither the global key Δ nor any other secret used in the original garbling.

This concludes the high-level overview of our main techniques. Naturally, several technical challenges arise when turning this into an actual construction, and we briefly highlight some of them (see Section 3.1 for more details). First, an observant reader may have noticed that when encrypting the authentication-tag function we encounter a form of key-dependent message (KDM) security: the parameter Δ appears both as part of the "secret key" and within the encrypted payload. Importantly, we do not require any additional circular-security assumptions. Instead, we prove that our scheme satisfies the required security property under the SIEDL assumption. (Circular security for encryption schemes based on subgroup-indistinguishability assumptions has previously been established in [BG10]). Second, the fact that the rerandomization matrices are triangular (with ones on the diagonal) ensures that the parameters inside the ciphertexts grow in a controlled and bounded manner. This property is crucial for the parameter analysis of the construction and for maintaining correctness across repeated rerandomization.

From Gates to Circuits. The jump from bilinear gates to full rerandomizable circuits is conceptually simple once we realize that we can interpret garbling as rerandomizing "default" canonical encryptions of the function.

Concretely, we first compile the circuit $\mathbf{C}$ into a deterministic "formatted" garbled circuit using identity randomness. This produces a canonical object $\mathsf{Gb}(\mathbf{C}; 1)$ that contains all gate structure but no effective randomness. A fully randomized garbling is then obtained by applying rerandomization:

$$\mathsf{Gb}(\mathbf{C}; \rho) \;=\; \mathsf{Rerand}(\rho,\, \mathsf{Gb}(\mathbf{C}; 1))\,.$$

In this sense, rerandomization is the central primitive of the construction. In practice, however, turning this idea into a full circuit construction requires substantial bookkeeping: wire randomness, ciphertext randomness, mask propagation, and authentication tags must all be updated consistently. While the high-level transition from gates to circuits is clean, ensuring that all invariants are preserved requires careful technical work, which we detail in Section 3.2.

[4] This matrix is the inverse transpose of the corresponding elements of R_{in} in the full construction.

Rerandomizable Garbling with Malicious Security. In many applications, each party in a rerandomization chain must verify that the previous rerandomization step was performed correctly. In our framework, this reduces to proving knowledge of the rerandomization randomness ρ such that

$$\boldsymbol{F'} = \mathsf{Rerand}(\rho, \boldsymbol{F}).$$

The algebraic structure of our construction makes this particularly natural. Since garbling is expressed as deterministic compilation followed by rerandomization, correctness of garbling reduces to correctness of rerandomization. Moreover, the modular composition law

$$\mathsf{Rerand}(\rho_1 \star \rho_2, C) = \mathsf{Rerand}(\rho_1, \mathsf{Rerand}(\rho_2, C))$$

ensures that rerandomization behaves homomorphically with respect to sequential updates. This structure enables black-box Σ-protocols for proving correct rerandomization and garbling, as we formalize in Appendix B.

2 Preliminaries

We refer the reader to Appendix A for additional preliminaries.

Notation. Throughout, we denote the computational security parameter by $\kappa \in \mathbb{N}$. Going forward, all function ensembles are indexed by $\kappa \in \mathbb{N}$ and all functions within them are indexed by $i \in \mathbb{N}$, though this notation may be omitted where this is implicit. Unless explicitly mentioned otherwise, all PPT algorithms take the security parameter 1^κ as implicit input, as well as the public parameters pp. "For all" statements are dependent on the public parameters, so if for all x a property $A(x)$ is said to hold perfectly or statistically, this means that x can be a (possibly inefficient) function of pp, while if $A(x)$ is a computational security property this means that it only holds when x is an efficient (PPT) function of pp. Let bold-face letters represent vectors, e.g. $\boldsymbol{v}$, where $\boldsymbol{v}_k$ refers to the element in the kth index of the vector $\boldsymbol{v}$ and $|\boldsymbol{v}|$ refers to the number of elements in $\boldsymbol{v}$. For an integer $n \in \mathbb{N}$ let $[n] := \{1, 2, \ldots, n\}$. Further for an $m < n, m \in \mathbb{N}$ let $[m : n] = [m, m+1, \cdots, n]$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if for all constants $c > 0$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$. We sometimes use $\epsilon(\cdot)$ to denote negligible functions. We denote by $\circ$ function composition. We also use the following group notation where $\star$ is the group operation and 1 denotes the identity element.

Definition 1 ([Sch97])**.** Let $\langle \mathbb{G}, \star \rangle$ be a group. A function $n \colon \mathbb{G} \to \mathbb{R}$ is a *G-seminorm* on $\mathbb{G}$ if it satisfies:

1. Nonnegativity: for all $g \in \mathcal{G}$, $n(g) \geq 0$.
2. Subadditivity: for all $g, g' \in \mathcal{G}$, $n(g \star g') \leq n(g) + n(g')$.
3. Symmetry: for all $g \in \mathcal{G}$, $n(g^{-1}) = n(g)$.
4. Identity: $n(1) = 0$.

We additionally require that the real numbers output by n can be efficiently represented and compared.

Circuit Notation. We represent a circuit $\mathbf{C} : \mathcal{X}^{\mathsf{n}} \to \mathcal{Y}^{\mathsf{m}}$ as a tuple of the form $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$ with $\mathsf{op}_g \in \{+, \times\}$ and $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$. Here Q represents the number of gates — we write it as a capital letter to keep it distinct from the order q of the easy subgroup $\mathbb{F}$ introduced in Section 2.2 — and n and m the input and output lengths, respectively. The circuit wires are numbered in topological order where those numbered in the first n are also the circuit input wires. Every wire beyond that is an output from some gate in the circuit. For each (binary) gate indexed $g \in [\mathsf{Q}]$, σ_1 and σ_2 represent the indices of its left and right input wires respectively, while $\sigma = \mathsf{n} + g$ represents the output wire index. op_g represents the *operation* performed by the gate. We write op rather than f here because a gate's operation

is merely a label in $\{+, \times\}$, and we want to keep it clearly distinct from the bilinear functions f that our building block actually garbles. Note that, in particular, the representation of $\mathbf{C}$ exceeds the number of gates. Looking ahead, this becomes important for garbling to work in polynomial time.

Arithmetic circuits with bounded integer computation. We use a model of arithmetic circuits consisting of addition, subtraction and (fan-in two) multiplication gates, all computed over $\mathbb{Z}$. We work in the bounded integer computation model, where there exists a bound $B = B(\kappa) \in \mathbb{N}$ that bounds the magnitude of circuit wire values. For a circuit $\mathbf{C}$ with n input wires, we say that an input vector $\boldsymbol{x} \in \mathbb{Z}^{\mathsf{n}}$ is admissible with respect to bound B if the inputs, outputs and every intermediate wire value obtained during the evaluation of $\mathbf{C}$ on $\boldsymbol{x}$ are in the range $[-B, B]$.

2.1 Garbling Schemes

Bellare et al. [BHR12] abstract a garbling scheme as a cryptographic object containing four algorithms as given in Definition 2. For a circuit $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g\in[\mathsf{Q}]})$, we denote the public information about $\mathbf{C}$ by $\phi(\mathbf{C})$, where ϕ is called a leakage function. In particular, $\phi_{\mathsf{circ}}(\mathbf{C}) = \mathbf{C}$ reveals the complete structure of the circuit, while $\phi_{\mathsf{topo}}(\mathbf{C}) = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma\}_{g\in[\mathsf{Q}]})$ reveals only the topology of the circuit, i.e. it does not leak the type of gates. Since the scope of this work is limited to schemes that garble gates in $\{+, \times\}$ and support free-addition, going forward, we use $\phi = \phi_{\mathsf{circ}}$.

Definition 2 (Garbling Scheme [BHR12])**.** Let pp be public parameters. For a bound $B \in \mathbb{N}$, let $\mathcal{F}$ be the family of functions whose computation can be represented by a circuit $\mathbf{C}$ in the B-bounded computation model, and whose inputs $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$ come from a space that is admissible with respect to B. For this function family, a garbling scheme is a tuple $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ of four polynomial-time algorithms (that take 1^κ and pp as implicit input) with the following syntax:

- $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}) \leftarrow \mathsf{Gb}(1^\kappa, \mathbf{C})$: on input a security parameter (in unary) and a circuit $\mathbf{C}$, returns a garbling $\boldsymbol{F}$, input encoding function $\boldsymbol{e}$, and output decoding function $\boldsymbol{d}$.
- $\boldsymbol{X} = \mathsf{En}(\boldsymbol{e}, \boldsymbol{x})$: returns the encoding $\boldsymbol{X}$ for function input $\boldsymbol{x}$.
- $\boldsymbol{Y} = \mathsf{Ev}(\boldsymbol{F}, \boldsymbol{X})$: returns the output labels $\boldsymbol{Y}$ by evaluating $\boldsymbol{F}$ on $\boldsymbol{X}$.
- $\{\perp, \boldsymbol{y}\} = \mathsf{De}(\boldsymbol{d}, \boldsymbol{Y})$: returns either the failure symbol $\perp$ or a value $\boldsymbol{y} = f(\boldsymbol{x})$.

These algorithms must satisfy the following properties:

- *Correctness*: For every circuit $\mathbf{C} : \mathcal{X}^{\mathsf{n}} \to \mathcal{Y}^{\mathsf{m}}$ and admissible input $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$,

$$\Pr[\boldsymbol{y} = \mathbf{C}(\boldsymbol{x}) : (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}) \leftarrow \mathsf{Gb}(\mathbf{C}),\ \boldsymbol{X} = \mathsf{En}(\boldsymbol{e}, \boldsymbol{x}),\ \boldsymbol{Y} = \mathsf{Ev}(\boldsymbol{F}, \boldsymbol{X}),\ \boldsymbol{y} = \mathsf{De}(\boldsymbol{d}, \boldsymbol{Y})] = 1$$

- *Privacy*: $\exists$ a PPT algorithm Sim such that for any circuit $\mathbf{C} : \mathcal{X}^{\mathsf{n}} \to \mathcal{Y}^{\mathsf{m}}$, and admissible input $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$,

$$\{\boldsymbol{F}, \boldsymbol{d}, \boldsymbol{X}\}_{(\boldsymbol{F},\boldsymbol{d},\boldsymbol{e})\leftarrow\mathsf{Gb}(\mathbf{C}),\boldsymbol{X}=\mathsf{En}(\boldsymbol{e},\boldsymbol{x})} \overset{\mathrm{c}}{\approx} \{\boldsymbol{F}, \boldsymbol{d}, \boldsymbol{X}\}_{(\boldsymbol{F},\boldsymbol{d},\boldsymbol{X})\leftarrow\mathsf{Sim}(1^\kappa,\phi(\mathbf{C}),\mathbf{C}(\boldsymbol{x}))}$$

- *Projective Encoding*: $\mathsf{En} : \mathcal{E} \times \mathcal{X}^{\mathsf{n}} \to \mathcal{Z}^{\mathsf{n}}$ is projective if: $\forall \boldsymbol{x} = \{x_i\}_{i\in[\mathsf{n}]}, \boldsymbol{x'} = \{x'_i\}_{i\in[\mathsf{n}]} \in \mathcal{X}^{\mathsf{n}}$, $\forall \boldsymbol{e} \in \mathcal{E}$, $\{\boldsymbol{X}_i\}_{i\in[\mathsf{n}]} = \mathsf{En}(\boldsymbol{e}, \boldsymbol{x})$ and $\{\boldsymbol{X'}_i\}_{i\in[\mathsf{n}]} = \mathsf{En}(\boldsymbol{e}, \boldsymbol{x'})$ such that $\forall i \in [\mathsf{n}]$, if $x_i = x'_i$ then $\boldsymbol{X}_i = \boldsymbol{X'}_i$.

Rerandomizable Garbling Schemes. We re-visit the definition of projective Rerandomizable Garbling Schemes [AHKP22] adapting it to function families for B-bounded integer computation.

Definition 3 (Rerandomizable Garbling Scheme)**.** For a bound $B \in \mathbb{N}$, let $\mathcal{F}$ be the family of functions whose computation can be represented by a circuit $\mathbf{C}$ in the B-bounded computation model, and whose inputs $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$ come from a space that is admissible with respect to B. For this function family, a garbling scheme $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ (Definition 2) is **T-rerandomizable** if there exists a PPT algorithm Rerand with the following properties:

- Let $\langle \mathsf{R}_{\mathsf{re}}, \star \rangle$ be a group with a G-seminorm n, where R_{re} is the (possibly infinite) set of rerandomizing randomnesses. Let 1 denote the identity element of R_{re}.

- Rerandomization must define a group action Upd, read "update". Whereas Rerand rerandomizes the garbling $\boldsymbol{F}$ alone and takes no private input, Upd is the derived map that packages the effect of Rerand on the *whole* triple: it applies a rerandomization to $\boldsymbol{F}$ and correspondingly updates the associated encoding and decoding information. In particular, Upd is not a second rerandomization procedure, and it does not replace Rerand.

$$\begin{array}{l}\underline{\mathsf{Upd}(\rho, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))}: \\ \quad (\boldsymbol{F'}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) = \mathsf{Rerand}(\rho, \boldsymbol{F}) \\ \quad \text{return} \begin{cases} \boldsymbol{F'} \\ \boldsymbol{e'} = \pi_{\mathsf{in}}(\boldsymbol{e}) \\ \boldsymbol{d'} = \pi_{\mathsf{out}}(\boldsymbol{d}) \end{cases}\end{array}$$

  Formally, $\forall \rho, \rho' \in \mathsf{R}_{\mathsf{re}}$, and any $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$ in the domains of the first arguments of Ev, En, and De, respectively,[5]

$$\mathsf{Upd}(1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$$

  and

$$\mathsf{Upd}(\rho' \star \rho, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = \mathsf{Upd}(\rho', \mathsf{Upd}(\rho, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))).$$

- There exists a PPT sampler $\mathsf{Gen}(1^\kappa)$ that samples $\rho \in \mathsf{R}_{\mathsf{re}}$ with $n(\rho) \leq 1$. We require that Gen have invertible sampling.

- There exists a (deterministic) efficient algorithm $\mathsf{Compile}$ that takes $\mathbf{C}$ as input and converts it into a valid input to Upd. The garbling algorithm Gb must be written as follows:

$$\begin{array}{l}\underline{\mathsf{Gb}(\mathbf{C})}: \\ \quad \rho \leftarrow \mathsf{Gen}(1^\kappa) \\ \quad (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}}) = \mathsf{Compile}(\mathbf{C}) \\ \quad (\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}') = \mathsf{Upd}(\rho, (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}})) \\ \quad \text{return } (\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}')\end{array}$$

  We write $\mathsf{Gb}(\rho, \mathbf{C}) = \mathsf{Upd}(\rho, \mathsf{Compile}(\mathbf{C}))$ for any $\rho \in \mathsf{R}_{\mathsf{re}}$ and $\mathbf{C} \in \mathcal{F}$.

- Rerand-*Correctness*: For all functions $\mathbf{C} \in \mathcal{F}$ and valid garbler randomnesses $\rho \in \mathsf{R}_{\mathsf{re}}$ where $n(\rho) \leq T$, the **correctness** property (Definition 2) holds with respect to $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}) = \mathsf{Gb}(\rho, \mathbf{C})$.

[5] Since we are in the rerandomizable setting, we cannot just ask $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$ to be in the support of Gb. As we apply noise for rerandomization, the parameters grow and leave the support of Gb.

- Rerand-*Privacy*: There exists PPT simulators $\mathsf{Sim}_0, \mathsf{Sim}_1$ such that for all functions $\mathbf{C} \in \mathcal{F}$, valid garbler randomnesses $\rho \in \mathsf{R}_{\mathsf{re}}$ where $n(\rho) \leq T$, and admissible inputs $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$:

$$\boxed{\begin{array}{l}\rho' \leftarrow \mathsf{Gen}(1^\kappa) \\ (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}) = \mathsf{Upd}(\rho' \star \rho, \mathsf{Compile}(\mathbf{C})) \\ \text{return } (\boldsymbol{F}, \mathsf{En}(\boldsymbol{e}, \boldsymbol{x}), \boldsymbol{d})\end{array}} \overset{\mathrm{c}}{\approx} \boxed{\begin{array}{l}(\boldsymbol{F}, \boldsymbol{X}, \mathsf{state}) \leftarrow \mathsf{Sim}_0(\rho) \\ \boldsymbol{d} \leftarrow \mathsf{Sim}_1(\mathsf{state}, \mathbf{C}(\boldsymbol{x})) \\ \text{return } (\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})\end{array}}$$

- *Masking*: There exists a PPT sampler $\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)$ where for all ρ in R_{re} with $n(\rho) \leq \beta$,

$$\{\rho' \star \rho\}_{\rho' \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)} \overset{\mathrm{s}}{\approx} \{\rho'\}_{\rho' \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)}.$$

  Here β is a bound on the *norm* of the rerandomization randomness being masked, and should not be confused with the bound B on the magnitude of the circuit's wire values. We require that $n(\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)) \leq \beta_{\mathrm{mask}} \cdot \beta$ when $\beta \geq 1$, where β_{mask} is sublinear in T.

In the above definition, note that Rerand-Correctness and Rerand-Privacy imply Correctness and Privacy (Definition 2) respectively, when ρ is the identity element.

Remark 4. A special case of the above definition that is the easiest to understand is when R_{re} is a finite group, and Gen is the uniform distribution on R_{re}. In this case, we can set $n(\rho) = 0$ for all ρ, and then Rerand-Correctness is equivalent to Correctness, Rerand-Privacy is equivalent to Privacy, and Masking holds with $\mathsf{Gen}_{\mathsf{mask}} = \mathsf{Gen}$. This special case is satisfied by the constructions of [AHKP22, HvdHJ25]. The extra complexity in our notational framework is due to the fact that we work over the integers, and we cannot sample uniformly from the integers.

2.2 Subgroup Indistinguishability

Our constructions are based on building-blocks whose instantiations are similar to the group-based constructions of additively homomorphic encryption for large messages, which require computing a discrete logarithmic to decrypt. We therefore need groups $\mathbb{G}$ with a message space subgroup $\mathbb{F}$ where discrete logarithm is easy, yet the messages can be hidden using a pseudorandom element of $\mathbb{G}$. We follow [BG10] in formalizing security as Subgroup Indistinguishability, where samples from a subgroup $\mathbb{H}$ of $\mathbb{G}$ must not be distinguishable from samples from the full group $\mathbb{G}$. Our definition is loosely inspired by the NIDLS framework [ADOS22], though we only need discrete logarithm and not distributed discrete logarithm. We adopt additive group notation for consistency with our constructions.

Definition 5. A group sampler $\mathcal{S}$ satisfies *subgroup indistinguishability with efficient discrete logarithms (SIEDL)* if it has the syntax

$$(\mathbb{G}, \mathbb{H}, \mathbb{F}, \mathsf{f}, q, \ell, \mathsf{DLog}) \leftarrow \mathcal{S}(1^\kappa),$$

where $\mathbb{F}$ and $\mathbb{H}$ are disjoint subgroups of the abelian group $\mathbb{G}$. $\mathbb{F}$ is cyclic, has order $|\mathbb{F}| = q$ and is generated by f. $\mathbb{H}$ must have order coprime to q. Each group output by $\mathcal{S}$ comes with an efficient algorithm (used implicitly by the sampling notation $\leftarrow$) to sample a statistically close to uniform element. Additionally, $\mathcal{S}$ must satisfy the following properties.

- **Efficient Discrete Logs in $\mathbb{F}$:** DLog is an efficient algorithm such that

$$x \equiv \mathsf{DLog}(x \cdot \mathsf{f}) \mod q \qquad \forall x \in \mathbb{Z}_q$$

- **ℓ-Bounded Keys:** For all $\mathsf{h} \in \mathbb{H}$,

$$\{x \cdot \mathsf{h}\}_{x \leftarrow [\ell]} \overset{\mathrm{c}}{\approx} \{\mathsf{h}'\}_{\mathsf{h}' \leftarrow \langle \mathsf{h} \rangle},$$

  where $\langle \mathsf{h} \rangle$ is the group generated by h.[6]

[6] This holds either statistically, with ℓ being sufficiently close to a multiple of $|\langle \mathsf{h} \rangle|$, or computationally, as a kind of small-exponent assumption.

- **Subgroup Indistinguishability:**

$$\{\mathsf{g}\}_{\mathsf{g}\leftarrow\mathbb{G}} \overset{\text{c}}{\approx} \{\mathsf{h}\}_{\mathsf{h}\leftarrow\mathbb{H}}$$

In ℓ-Bounded Keys and Subgroup Indistinguishability, all outputs of $\mathcal{S}$ are implicitly given to the distinguisher.

We prove the following lemma in Appendix C.

Lemma 6. *When* $\mathsf{pp} = (\mathbb{G}, \mathbb{H}, \mathbb{F}, \mathsf{f}, q, \ell, \mathsf{DLog}) \leftarrow \mathcal{S}(1^\kappa)$ *are sampled as a SIEDL group (Definition 5), for all* $y, y' \in \mathbb{Z}_q$ *it holds that*

$$\{\mathsf{pp}, \mathsf{h}\}_{\mathsf{h}\leftarrow\mathbb{H}} \overset{\text{c}}{\approx} \{\mathsf{pp}, \mathsf{h} + y \cdot \mathsf{f}\}_{\mathsf{h}\leftarrow\mathbb{H}}$$

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}\}_{\mathsf{h}\leftarrow\mathbb{H}, x\leftarrow[\ell]} \overset{\text{c}}{\approx} \{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h} + y \cdot \mathsf{f}\}_{\mathsf{h}\leftarrow\mathbb{H}, x\leftarrow[\ell]}$$

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h}, x' \cdot x \cdot \mathsf{h}\}_{\substack{\mathsf{h}\leftarrow\mathbb{H}, \\ x,x'\leftarrow[\ell]}} \overset{\text{c}}{\approx} \{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + y \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + y' \cdot \mathsf{f}\}_{\substack{\mathsf{h}\leftarrow\mathbb{H}, \\ x,x'\leftarrow[\ell]}}.$$

For our construction, it is important that q/ℓ can be made exponential. We will discuss how to instantiate a SIEDL group satisfying this, from either DCR or class groups. Other possible instantiations are from Naccache and Stern's modification of the Benaloh cryptosystem [Ben87, NS98] and from the Joye–Libert cryptosystem [BHJL17] (and from its modification in [ADOS22]).

Instantiating a SIEDL Group from DCR. Let $\mathcal{S}$ sample an RSA modulus $N = p' \cdot q'$, with p' and q' are distinct primes of the same bit length. $\mathbb{G}$ is then the Damgård–Jurik group $\mathbb{Z}^\times_{N^{\zeta+1}}$, $\mathbb{H}$ is the subgroup of perfect powers of N^ζ, and $\mathbb{F}$ is the subgroup of elements equal to 1 modulo N [DJ01]. The Subgroup Indistinguishability property is equivalent to the DCR assumption [DJ01]. $\mathbb{F}$ has order $q = N^\zeta$, is generated by $f = \sum_{i=0}^{\zeta} \frac{N^i}{i!}$, and has an efficient DLog algorithm by evaluating the power series for natural logarithm. $\mathbb{H}$ has order $\varphi(N) = (p'-1)(q'-1)$, which is coprime to N (and so q) because p' and q' have the same bit length (neither can be less than half of the other). ℓ-Bounded Keys hold statistically with $\ell = N$ because N is very close to $\varphi(N)$: $\frac{N-\varphi(N)}{N} = \frac{p'+q'-1}{N} = O(N^{-1/2})$ is negligible. The ratio q/ℓ is then $N^{\zeta-1}$.

Instantiating a SIEDL Group from Class Groups. In the class group encryption scheme of [CL15], one chooses a discriminant $\Delta_K = -pq \equiv 1 \mod 4$ (for public primes p, q), and a discriminant $\Delta_q = q^2\Delta_K$ for an order $\mathcal{O}_{\Delta_q}$, and sets $\mathbb{G}$ to be the class group of $\mathcal{O}_{\Delta_q}$.[7] Then there is a public element f in the class group of $\mathcal{O}_{\Delta_q}$ that is generates a subgroup $\mathbb{F}$ of order q, and there is an efficient discrete logarithm in this subgroup. We follow [ADOS22] in assuming that the class group of $\mathcal{O}_{\Delta_K}$ is indistinguishable from whole group $\mathbb{G}$, and so making this subgroup be $\mathbb{H}$. The size of $\mathbb{H}$ will be roughly $\sqrt{|\Delta_K|} = \sqrt{pq}$, so we can either set $\ell = 2^\kappa\sqrt{pq}$ for statistical security of ℓ-Bounded Keys, or use a small exponent assumption and set $\ell = 2^{2\kappa}$. The Cohen–Lenstra heuristic indicates $|\mathbb{H}|$ should be coprime to q except with negligible probability [CL15].

The settings proposed in main scheme of [CL15] chooses $p > 4q$ to make the discrete logarithm algorithm simpler. In this setting, $|\mathbb{H}| \approx \sqrt{pq} > 2q$, so we would have to rely on a small-exponent assumption to set $\ell = 2^{2\kappa}$ so that q/ℓ can be exponential. However, the extension in [CL15, Section 4] allows setting p and q independently, and in this setting $q/\ell = 2^{-\kappa}\sqrt{q/p}$ can be made exponential.

3 Rerandomizable Arithmetic Garbling Scheme

In this section we construct constant-rate rerandomizable garbling schemes for B-bounded arithmetic circuits (Definition 3). We base these on SIEDL groups (Definition 5) sampled such

[7] Note that p and q are swapped in our notation compared with [CL15], as we follow the notation of [ADOS22].

that the order of the subgroup $\mathbb{F}$ is much greater than the bound B. We tackle the problem of constructing our garbling scheme in the following stages:

- First, we define a function family $\mathcal{F}_{\mathsf{pp}}$ of bilinear functions over such an SIEDL group (Definition 5) where pp contains the group description.
- We then define and construct an *updatable* garbling scheme for $\mathcal{F}_{\mathsf{pp}}$. Informally, this is an object that allows one to evaluate a private input on a secret function from this family, in such a way that only the output is revealed. Further, the function garbling itself can be updated to a garbling of an affine function of the original. Details for this can be found in Section 3.1.
- We use this object in a black-box way to construct rerandomizable garbling schemes for B-bounded arithmetic circuits. The high-level idea is to implement the multiplication gates in the circuit with this and leverage its linear properties to allow for free addition gates. This is detailed in Section 3.2.
- Finally, in Appendix B we present efficient Σ-protocols for proving correctness of garbling and rerandomizing, that can by converted into zero-knowledge proofs-of-knowledge using well-known techniques in the literature.

We start by defining below the family of bilinear functions which forms the basis for defining our building-blocks.

Definition 7. For a bound $B \in \mathbb{N}$, let $\mathsf{pp}' = (\mathbb{G}, \mathbb{H}, \mathbb{F}, \mathsf{f}, q, \ell, \mathsf{DLog}) \leftarrow \mathcal{S}(1^\kappa)$ be a sampler for an SIEDL group (Definition 5) where $q > [2^{2\kappa} B\ell]$. The PPT algorithm $\mathsf{Setup}(1^\kappa) \to \mathsf{pp} = (g, \mathcal{X}, \mathcal{Y}, \mathsf{pp}')$ where:

$$\mathsf{pp}' = (\mathbb{G}, \mathbb{H}, \mathbb{F}, \mathsf{f}, q, \ell, \mathsf{DLog}) \leftarrow \mathcal{S}(1^\kappa) \qquad g \leftarrow \mathbb{H} \qquad \mathcal{X} = \mathbb{Z} \qquad \mathcal{Y} = \mathbb{Z}_q$$

We define $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ as the family of bilinear functions over the group parameterized by pp', i.e. each $f \in \mathcal{F}_{\mathsf{pp}}$ can be represented as

$$f(x_0, x_1) = \begin{bmatrix} 1 & x_0 \end{bmatrix} \boldsymbol{F} \begin{bmatrix} 1 \\ x_1 \end{bmatrix} \text{ where } \boldsymbol{F} \in \mathbb{Z}_q^{2\times 2}$$

In the following, we define and instantiate an object that we call an updatable garbling scheme for the function family $\mathcal{F}_{\mathsf{pp}}$ given in Definition 7.

3.1 Updatable Garbling Scheme for Bilinear Functions

In this section, we first define an updatable garbling scheme for the family of bilinear functions $\mathcal{F}_{\mathsf{pp}}$ (Definition 7). We formalize this as a tuple of six algorithms where input encoding is inherently projective, as detailed in Definition 8. The high-level idea involves viewing the complete space of garbling randomness R as a group with two nested sub-groups: R_{pk}, the space of (global) randomness from which elements common to both input wires are sampled; and $\mathsf{R}_{\mathsf{MAC}}$ the space of (wire) randomness from which elements unique to each input wire is sampled. The scheme is parameterized by a default global key pk^* and wire key pk_w.

Definition 8 (Updatable Garbling Scheme)**.** For a function family $\mathcal{F}_{\mathsf{pp}} \subseteq \mathcal{X}^2 \to \mathcal{Y}$ (Definition 7), where $\mathcal{X}$ and $\mathcal{Y}$ are abelian groups written additively, and input bounds $\mathcal{X}' \subseteq \mathcal{X}$, an **updatable garbling scheme** is a tuple of PPT algorithms $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ that take 1^κ and pp as implicit input and have the following properties:

- Let $\langle \mathsf{R}, \star \rangle$ be a group where R is the (possibly infinite) set of ciphertext transformations used for rerandomization, and let $n \colon \mathsf{R} \to \mathbb{R}$ be a G-seminorm on R. We require that R is structured as follows:

- R contains two subgroups $\mathsf{R}_{\mathsf{MAC},1}, \mathsf{R}_{\mathsf{MAC},2}$, representing transformations of the first and second input wires, respectively. $\mathsf{R}_{\mathsf{MAC},1} \cong \mathsf{R}_{\mathsf{MAC},2}$, so we will sometimes omit the subscript and write $\mathsf{R}_{\mathsf{MAC}}$.
- R contains a subgroup R_{pk} of global public key transformations. We require R_{pk} to be the intersection $\mathsf{R}_{\mathsf{MAC},1} \cap \mathsf{R}_{\mathsf{MAC},2}$.
- $\mathsf{R}_{\mathsf{MAC},\sigma}$ contains commutative normal subgroups $\overline{\mathsf{R}}_{\mathsf{MAC},\sigma} \trianglelefteq \mathsf{R}_{\mathsf{MAC},\sigma}$ (for $\sigma \in [2]$) of local wire transformations (leaving the global key unchanged). We also require that $\overline{\mathsf{R}}_{\mathsf{MAC},1}$ and $\overline{\mathsf{R}}_{\mathsf{MAC},2}$ commute with each other, so $\overline{\mathsf{R}}_{\mathsf{MAC},1} \star \overline{\mathsf{R}}_{\mathsf{MAC},2}$ is a commutative subgroup of R.
- R contains a commutative normal subgroup $\overline{\mathsf{R}} \trianglelefteq \mathsf{R}$ of local ciphertext transformations (leaving the input wires unchanged)
- Every $\rho_c \in \mathsf{R}$ can be written uniquely as $\overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta$ where $\overline{\rho}_c \in \overline{\mathsf{R}}_{\mathsf{MAC}}$, $\{\overline{\rho}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}\}_{\sigma\in[2]}$, and $\Delta \in \mathsf{R}_{\mathsf{pk}}$. Equivalently, $\mathsf{R}_{\mathsf{MAC},\sigma}$ is the semidirect product $\overline{\mathsf{R}}_{\mathsf{MAC},\sigma} \rtimes \mathsf{R}_{\mathsf{pk}}$, and R is the semidirect product $\overline{\mathsf{R}} \rtimes \langle \mathsf{R}_{\mathsf{MAC},1} \cup \mathsf{R}_{\mathsf{MAC},2} \rangle$. This decomposition induces homomorphisms $\mathsf{key}\colon \mathsf{R} \to \mathsf{R}_{\mathsf{pk}}$ and $\mathsf{in}_\sigma\colon \mathsf{R} \to \mathsf{R}_{\mathsf{MAC},\sigma}$, where

$$\mathsf{key}(\overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta) = \Delta$$
$$\mathsf{in}_\sigma(\overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta) = \overline{\rho}_\sigma \star \Delta,$$

  extracting the global key transformation, or the transformation for the input wire σ, respectively. These homomorphisms and decomposition must respect the G-seminorm:

$$n(\mathsf{key}(\rho_c)) \le n(\rho_c) \qquad n(\mathsf{in}_\sigma(\rho_c)) \le n(\rho_c) \qquad n(\overline{\rho}_1 \star \overline{\rho}_2 \star \Delta) \le n(\rho_c).$$

- Let $\mathsf{offset}\colon \mathsf{R}_{\mathsf{MAC}} \to \mathcal{X}$ be the homomorphism extracting the offset $\boldsymbol{m}_\sigma \in \mathcal{X}$ from a wire homomorphism $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$. We require R_{pk} to be in the kernel of offset. We denote by

$$\overline{\mathsf{R}}_{\mathsf{MAC},\sigma}{}^{\boldsymbol{m}_\sigma} = \{\overline{\rho}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma} \mid \mathsf{offset}(\overline{\rho}_\sigma) = \boldsymbol{m}_\sigma\}$$

  the subset of $\overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$ with offset $\boldsymbol{m}_\sigma$.

- The set of ciphertexts CTXT must structured as $\mathsf{PK}_1 \times_{\mathsf{PK}} \mathsf{PK}_2 \times \overline{\mathsf{CTXT}}$ and wire public key set $\mathsf{PK}_{\mathsf{MAC}}$ must be structured as $\mathsf{PK}_{\mathsf{MAC}} = \mathsf{PK} \times \overline{\mathsf{PK}_{\mathsf{MAC}}}$. That is, the elements of $\mathsf{PK}_{\mathsf{MAC}}$ are $(\mathsf{pk}, \overline{\mathsf{pk}}_\sigma)$ for $\mathsf{pk} \in \mathsf{PK}$ and $\overline{\mathsf{pk}}_\sigma \in \overline{\mathsf{PK}_{\mathsf{MAC}}}$, while elements of CTXT are $(\mathsf{pk}_1, \mathsf{pk}_2, \overline{c})$ for $\mathsf{pk}_1 \in \mathsf{PK}_1, \mathsf{pk}_2 \in \mathsf{PK}_2, \overline{c} \in \overline{\mathsf{CTXT}}$ such that pk_1 and pk_2 contain the same pk.

- Let Rerand be a group action of R on ciphertexts CTXT. We require that Rerand be structured as follows:

$\underline{\mathsf{Rerand}(\rho_c, c)}$:	$\underline{\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\rho_\sigma = \overline{\rho}_\sigma \star \Delta, \mathsf{pk}_\sigma)}$:
parse $c = (\mathsf{pk}_1, \mathsf{pk}_2, \overline{c})$	parse $\mathsf{pk}_\sigma = (\mathsf{pk}, \overline{\mathsf{pk}}_\sigma)$
$\mathsf{pk}'_\sigma = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\mathsf{in}_\sigma(\rho_c), \mathsf{pk}_\sigma)$ for $\sigma \in [2]$	$\mathsf{pk}' = \mathsf{Re}_{\mathsf{pk}}(\Delta, \mathsf{pk})$
$\overline{c}' = \overline{\mathsf{Rerand}}(\rho_c, c)$	$\overline{\mathsf{pk}}'_\sigma = \overline{\mathsf{Re}_{\mathsf{pk\text{-}wire}}}(\overline{\rho}_\sigma, \mathsf{pk}', \overline{\mathsf{pk}}_\sigma)$
return $(\mathsf{pk}'_1, \mathsf{pk}'_2, \overline{c}')$	return $(\mathsf{pk}', \overline{\mathsf{pk}}'_\sigma)$

  for some PPT algorithms $\mathsf{Re}_{\mathsf{pk}}$, $\overline{\mathsf{Re}_{\mathsf{pk\text{-}wire}}}$, and $\overline{\mathsf{Rerand}}$. Note that this implies that $\mathsf{Re}_{\mathsf{pk}}$ is a group action of R_{pk} on global public keys PK, and $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$ is a group action of $\mathsf{R}_{\mathsf{MAC}}$ on wire public keys $\mathsf{PK}_{\mathsf{MAC}}$.

- The wire labels are pairs $(\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)$, where $\overline{\boldsymbol{x}}_\sigma \in \mathcal{X}$ is the offsetted input and $\boldsymbol{X}_\sigma \in \mathcal{M}$ is the MAC authenticating the input $\boldsymbol{x}_\sigma$, generated using an algorithm $\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)$. $\mathcal{M}$

must be an abelian group, written additively. offset and MAC must define a group action $\mathsf{Re}_{\mathsf{MAC}} \colon \mathsf{R}_{\mathsf{MAC}} \times (\mathcal{X} \times \mathcal{M}) \to (\mathcal{X} \times \mathcal{M})$:

$$\boxed{\begin{array}{l}\underline{\mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)):} \\ \quad \overline{\boldsymbol{x}}'_\sigma = \overline{\boldsymbol{x}}_\sigma + \mathsf{offset}(\rho_\sigma) \\ \quad \boldsymbol{X}'_\sigma = \boldsymbol{X}_\sigma + \mathsf{MAC}(\rho_\sigma, \overline{\boldsymbol{x}}_\sigma) \\ \quad \text{return } (\overline{\boldsymbol{x}}'_\sigma, \boldsymbol{X}'_\sigma)\end{array}}$$

- The wire public keys $\overline{\mathsf{PK}_{\mathsf{MAC}}}$ must form an abelian group, as must the ciphertexts $\overline{\mathsf{CTXT}}$. If $\mathsf{pk}_\sigma = (\mathsf{pk}, \overline{\mathsf{pk}}_\sigma) \in \mathsf{PK}_{\mathsf{MAC}}$ and $\mathsf{pk}'_\sigma = (\mathsf{pk}, \overline{\mathsf{pk}}'_\sigma) \in \mathsf{PK}_{\mathsf{MAC}}$ contain the same global public key, then we write
$$\mathsf{pk}_\sigma + \mathsf{pk}'_\sigma = (\mathsf{pk}, \overline{\mathsf{pk}}_\sigma + \overline{\mathsf{pk}}'_\sigma).$$
Similarly, if ciphertexts $c, c' \in \mathsf{CTXT}$ contain the same wire public keys pk_1 and pk_2, we write $c + c'$ to denote adding the corresponding $\overline{c}$ and $\overline{c}'$, while leaving the public keys unchanged. We require $\mathsf{Compile}$ to be linear, i.e., it must be a group homomorphism from $\mathcal{F}_{\mathsf{pp}}$ into ciphertexts

- For every global public key $\mathsf{pk} \in \mathsf{PK}$, there must be an subgroup $\mathcal{W}_{\mathsf{pk}} \subseteq \overline{\mathsf{PK}_{\mathsf{MAC}}} \times \mathcal{X} \times \mathcal{M}$ of valid wires $(\overline{\mathsf{pk}}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)$. We write
$$\mathcal{W} = \left\{((\mathsf{pk}, \overline{\mathsf{pk}}_\sigma), \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \mid \mathsf{pk} \in \mathsf{PK}, (\overline{\mathsf{pk}}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \in \mathcal{W}_{\mathsf{pk}}\right\} \subseteq \mathsf{PK}_{\mathsf{MAC}} \times \mathcal{X} \times \mathcal{M}.$$

- $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$ and $\mathsf{Re}_{\mathsf{MAC}}$ must define a group action on valid wires: for all $\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma$,
$$(\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \in \mathcal{W} \Rightarrow (\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\rho_\sigma, \mathsf{pk}_\sigma), \mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma))) \in \mathcal{W}.$$

- Evaluation must be linear in c, i.e., if c and c' contain the same wire public keys then for all inputs $\overline{\boldsymbol{x}} \in \overline{X}^2$ and input MACs $(\boldsymbol{X}_\sigma)_{\sigma \in [2]}$,
$$\mathsf{Eval}(c + c', (\boldsymbol{X}_\sigma)_{\sigma \in [2]}, \overline{\boldsymbol{x}}) = \mathsf{Eval}(c, (\boldsymbol{X}_\sigma)_{\sigma \in [2]}, \overline{\boldsymbol{x}}) + \mathsf{Eval}(c', (\boldsymbol{X}_\sigma)_{\sigma \in [2]}, \overline{\boldsymbol{x}}).$$
whenever c and c' both decrypt successfully (i.e., don't return $\perp$).

- The group action Rerand must be linear in c and $\overline{\rho}_c$, for any $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$. In detail, if c and c' contain the same wire public keys, then for all $\{\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}\}_{\sigma \in [2]}$, and for all $\overline{\rho}_c, \overline{\rho}'_c \in \overline{\mathsf{R}}$,
$$\mathsf{Rerand}(\overline{\rho}_c \star \overline{\rho}'_c \star \rho_1 \star \rho_2, c + c') = \mathsf{Rerand}(\overline{\rho}_c \star \rho_1 \star \rho_2, c) + \mathsf{Rerand}(\overline{\rho}'_c \star \rho_1 \star \rho_2, c').$$

- The wire public key group action $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$ must be linear in pk_σ and $\overline{\rho}_\sigma$, for any $\Delta \in \mathsf{R}_{\mathsf{pk}}$. In detail, if pk_σ and pk'_σ contain the same global public key pk, then for all $\Delta \in \mathsf{R}_{\mathsf{pk}}$, and for all $\overline{\rho}_\sigma, \overline{\rho}'_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$,
$$\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \overline{\rho}'_\sigma \star \Delta, \mathsf{pk}_\sigma + \mathsf{pk}'_\sigma) = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \Delta, \mathsf{pk}_\sigma) + \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}'_\sigma \star \Delta, \mathsf{pk}'_\sigma).$$
Similarly, the MAC group action $\mathsf{Re}_{\mathsf{MAC}}$ must be linear in $\overline{\boldsymbol{x}}_\sigma$, $\boldsymbol{X}_\sigma$, and $\overline{\rho}_\sigma$, for any $\Delta \in \mathsf{R}_{\mathsf{pk}}$. Using the structure of $\mathsf{Re}_{\mathsf{MAC}}$ to write the condition in terms of MAC, it is that for all $\boldsymbol{x}_\sigma, \boldsymbol{x}'_\sigma \in \mathcal{X}$, all $\Delta \in \mathsf{R}_{\mathsf{pk}}$, and for all $\overline{\rho}_\sigma, \overline{\rho}'_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$,
$$\mathsf{MAC}(\overline{\rho}_\sigma \star \overline{\rho}'_\sigma \star \Delta, \overline{\boldsymbol{x}}_\sigma + \overline{\boldsymbol{x}}'_\sigma) = \mathsf{MAC}(\overline{\rho}_\sigma \star \Delta, \overline{\boldsymbol{x}}_\sigma) + \mathsf{MAC}(\overline{\rho}'_\sigma \star \Delta, \overline{\boldsymbol{x}}'_\sigma).$$

- Let $\mathcal{G}_{\mathsf{offset}}, \mathcal{G}_{\mathsf{pk}}, \mathcal{G}_{\mathsf{MAC}}, \mathcal{G}_{\mathsf{Enc}}$ be PPT samplers with invertible sampling that respectively sample $\boldsymbol{m}_\sigma \in \mathcal{X}$, $\Delta \in \mathsf{R}_{\mathsf{pk}}$, $\overline{\rho}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC}}^{\boldsymbol{m}_\sigma}$, and $\overline{\rho}_c \in \overline{\mathsf{R}}$. We denote by $\mathcal{G}(1^\kappa)$ the combined sampler:

$\underline{\mathcal{G}(1^\kappa)}$:
$\boldsymbol{m}_\sigma \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa)$ for $\sigma \in [2]$
$\Delta \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)$
$\overline{\rho}_\sigma \leftarrow \mathcal{G}_{\mathsf{MAC}}(1^\kappa, \boldsymbol{m}_\sigma)$ for $\sigma \in [2]$
$\overline{\rho}_c \leftarrow \mathcal{G}_{\mathsf{Enc}}(1^\kappa)$
return $\overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta$

  It must satisfy $n(\mathcal{G}(1^\kappa)) \le 1$.

- *Correctness*: Let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be public default values, where $\mathsf{pk}_w = (\mathsf{pk}^*, 0) \in \mathsf{PK}_{\mathsf{MAC}}$ and 0 is the identity element in $\overline{\mathsf{PK}_{\mathsf{MAC}}}$. Let $\mathsf{Compile}$ be the deterministic "encryption" function, mapping a function f into the ciphertext space; we require that $\mathsf{Compile}$ is a homomorphism and that $\mathsf{Compile}(f) = (\mathsf{pk}_w, \mathsf{pk}_w, \overline{\mathsf{Compile}}(f))$. For every $f \in \mathcal{F}_{\mathsf{pp}}$, input $\boldsymbol{x} \in \mathcal{X}^2$, offset $\boldsymbol{m} \in \mathcal{X}^2$, and randomness $\rho_c = \overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta$ for $\overline{\rho}_c \in \overline{\mathsf{R}}$, $\{\rho_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}^{\boldsymbol{m}_\sigma}\}_{\sigma\in[2]}$, $\Delta \in \mathsf{R}_{\mathsf{pk}}$, it holds that,

$$\Pr\left[\begin{array}{l} y = f(\boldsymbol{x}) \ : c = \mathsf{Compile}(f)\ ;\ \{\boldsymbol{X}_\sigma = \mathsf{MAC}(\overline{\rho}_\sigma \star \Delta, \boldsymbol{x}_\sigma)\}_{\sigma\in[2]} \\ \qquad\qquad c' = \mathsf{Rerand}(\rho_c, c)\ ;\ y = \mathsf{Eval}(c', (\boldsymbol{X}_\sigma)_{\sigma\in[2]}, \boldsymbol{x} + \boldsymbol{m}) \end{array}\right] = 1$$

- $\mathsf{R}_{\mathsf{MAC},\sigma}$ must be in the stabilizer subgroup of $\mathsf{Compile}(1)$, so that deterministic encryption of a constant function 1 does not change when its input wires are rerandomized. That is, for all $\sigma \in [2]$ and $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$,

$$\overline{\mathsf{Compile}}(1) = \overline{\mathsf{Rerand}}(\rho_\sigma, \mathsf{Compile}(1)).$$

- *Input Masking*: For all $x \in \mathcal{X}'$,

$$\{m + x\}_{m\leftarrow\mathcal{G}_{\mathsf{offset}}(1^\kappa)} \overset{\mathrm{s}}{\approx} \{m\}_{m\leftarrow\mathcal{G}_{\mathsf{offset}}(1^\kappa)}.$$

- *Masking*: There exists a PPT samplers $\mathcal{G}^*_{\mathsf{pk}}$, $\mathcal{G}^*_{\mathsf{MAC}}$, and $\mathcal{G}^*_{\mathsf{Enc}}$ such that for all $\Delta \in \mathsf{R}_{\mathsf{pk}}, \overline{\rho}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}, \overline{\rho}_c \in \overline{\mathsf{R}}$, each with G-seminorm upper bounded by β,

$$\begin{aligned} \{\Delta' \star \Delta\}_{\Delta'\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{\mathrm{s}}{\approx} \{\Delta'\}_{\Delta'\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))} \\ \{\overline{\rho'}_\sigma \star \overline{\rho}_\sigma\}_{\overline{\rho'}_\sigma\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{\mathrm{s}}{\approx} \{\overline{\rho'}_\sigma\}_{\overline{\rho'}_\sigma\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))} \\ \{\overline{\rho'}_c \star \overline{\rho}_c\}_{\overline{\rho'}_c\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{\mathrm{s}}{\approx} \{\overline{\rho'}_c\}_{\overline{\rho'}_c\leftarrow\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))}. \end{aligned}$$

  For some bound $\beta_{\mathrm{mask}} = 2^{\mathrm{poly}(\kappa)}$, the output sampler outputs must have G-seminorm upper bounded by $\beta_{\mathrm{mask}} \cdot \beta$ when $\beta \ge 1$.

While we only defined masking for each component in the R decomposition, if you combine them together then you can rerandomize the whole group element. This is not completely straightforward due to non-commutativity. We prove the following lemma in Appendix D.2.

Lemma 9. *For all ρ_c in R with $n(\rho_c) \le \beta$,*

$$\{\rho'_c \star \rho_c\}_{\rho'_c\leftarrow\mathsf{Gen}_{\mathsf{mask}}(1^\kappa,\beta)} \overset{\mathrm{s}}{\approx} \{\rho'_c\}_{\rho'_c\leftarrow\mathsf{Gen}_{\mathsf{mask}}(1^\kappa,\beta))},$$

where

$\underline{\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)}$:
$\Delta \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa, \beta)$
$\overline{\rho}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{MAC}}(1^\kappa, 2(\beta_{\mathrm{mask}} + 1)\beta)$ for $\sigma \in [2]$
$\overline{\rho}_c \leftarrow \mathcal{G}^*_{\mathsf{Enc}}(1^\kappa, (8\beta^2_{\mathrm{mask}} + 10\beta_{\mathrm{mask}} + 2)\beta)$
return $\overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta$

Homomorphic evaluation of affine functions. Note that the group structure and linearity of updatable garbling schemes allows us to homomorphically 'update' the function being garbled. More precisely, this garbling scheme supports affine updates on functions: if $\{c_i\}_{i\in[n]}$ are ciphertexts such that each c_i is a garbling of $f_i \in \mathcal{F}_{\mathsf{pp}}$ under the same keys, and $g(x_1,\ldots,x_n) = g_0 + \sum_i g_i x_i$ is an affine function over n variables, then it is possible to homomorphically generate a ciphertext c from $\{c_i\}_{i\in[n]}$ such that c encrypts the function $g(\{f_i(x)\}_{i\in[n]})$:

$$c = (\mathsf{pk}_1, \mathsf{pk}_2, \overline{\mathsf{Compile}}(g_0)) + \sum_i g_i \cdot c_i,$$

where pk_1 and pk_2 are the input wire public keys in the ciphertexts c_i. We write affine evaluation on ciphertexts as simply $g(c_1,\ldots,c_n)$.

Definition 10 (KDM Security of Updatable Garbling)**.** For a function family $\mathcal{F}_{\mathsf{pp}} \subseteq \mathcal{X}^2 \to \mathcal{Y}$ (Definition 7), and input bounds $\mathcal{X}' \subseteq \mathcal{X}$, let $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definition 8). Let $\mathcal{F}^*_{\mathsf{pp}} : \mathsf{R}_{\mathsf{pk}} \to \mathcal{F}_{\mathsf{pp}}$ be a space of key-dependent function. For a bound T, UGS is T-secure for $\mathcal{F}^*_{\mathsf{pp}}$ if the following properties hold:

- *Ciphertext Simulation*: There exists a PPT simulator $\mathcal{S}_{\mathsf{Enc}}$ such that for all KDM functions $f \in \mathcal{F}^*_{\mathsf{pp}}$, all inputs $\boldsymbol{x} \in \mathcal{X}^2$, and all transformations $\rho_c \in \mathsf{R}$:

$$\boxed{\begin{array}{l}\overline{\rho}'_c \leftarrow \mathcal{G}_{\mathsf{Enc}}(1^\kappa)\\ c = \mathsf{Rerand}(\overline{\rho}'_c \star \rho_c, \mathsf{Compile}(f(\Delta)))\\ \text{return } c\end{array}} \overset{\mathrm{c}}{\approx} \boxed{\begin{array}{l}\text{for } \sigma \in [2], \begin{cases}\mathsf{pk}_\sigma & = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\rho_\sigma, \mathsf{pk}_w)\\ (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) & = \mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\boldsymbol{x}_\sigma, 0))\end{cases}\\ c \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, f(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f)\\ \text{return } c\end{array}}$$

  where $\Delta = \mathsf{key}(\rho_c)$ and $\rho_\sigma = \mathsf{in}_\sigma(\rho_c)$ for $\sigma \in [2]$.

- *MAC Simulation*: There exists a PPT simulators $\mathcal{S}_{\mathsf{MAC}}$ such that for all wire transformations $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$ with $n(\rho_\sigma) \le T$, possible global key rerandomizations $\Delta' \in \mathit{supp}(\mathcal{G}_{\mathsf{pk}}(1^\kappa))$, possible offsets $\boldsymbol{m}'_\sigma \in \mathit{supp}(\mathcal{G}_{\mathsf{offset}}(1^\kappa))$, and all valid inputs $\boldsymbol{x}_\sigma \in \mathcal{X}'$:

$$\boxed{\begin{array}{l}\overline{\rho}'_\sigma \leftarrow \mathcal{G}_{\mathsf{MAC}}(\boldsymbol{m}'_\sigma)\\ \\ \mathsf{pk}_\sigma = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}'_\sigma \star \Delta' \star \rho_\sigma, \mathsf{pk}_w)\\ X_\sigma = \mathsf{MAC}(\overline{\rho}'_\sigma \star \Delta' \star \rho_\sigma, \boldsymbol{x}_\sigma)\\ \text{return } (\mathsf{pk}_\sigma, X_\sigma)\end{array}} \overset{\mathrm{s}}{\approx} \boxed{\begin{array}{l}\Delta = \mathsf{key}(\rho_\sigma)\\ \mathsf{pk} = \mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)\\ \overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)\\ (\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)\\ \text{return } (\mathsf{pk}_\sigma, X_\sigma)\end{array}}$$

  We require that the $(\mathsf{pk}_\sigma, X_\sigma)$ output by $\mathcal{S}_{\mathsf{MAC}}$ always define valid wires $(\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, X_\sigma) \in \mathcal{W}$, like how $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$ and MAC always output valid wires.

- *Privacy*: There exists a PPT simulator Sim such that for all global keys $\Delta \in \mathsf{R}_{\mathsf{pk}}$ and all PPT adversaries $\mathcal{A}$,

$$\boxed{\begin{array}{l}\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)\\ \mathsf{pk} = \mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)\\ (\mathsf{state}, \mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{x}, y, f) \leftarrow \mathcal{A}(\mathsf{pk})\\ \text{abort if } \exists \sigma \in [2].\ (\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \notin \mathcal{W}\\ c \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, y, \boldsymbol{x}, f)\\ \text{return } (c, \mathsf{state})\end{array}} \overset{\mathrm{c}}{\approx} \boxed{\begin{array}{l}\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)\\ \mathsf{pk} = \mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)\\ (\mathsf{state}, \mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{x}, y, f) \leftarrow \mathcal{A}(\mathsf{pk})\\ \text{abort if } \exists \sigma \in [2].\ (\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \notin \mathcal{W}\\ c \leftarrow \mathsf{Sim}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, y)\\ \text{return } (c, \mathsf{state})\end{array}}$$

  where $\mathcal{A}$ must output $\mathsf{pk}_\sigma = (\mathsf{pk}, \overline{\mathsf{pk}}_\sigma)$, $\boldsymbol{x} \in \mathcal{X}^2$, $y \in \mathcal{Y}$, and $f \in \mathcal{F}^*_{\mathsf{pp}}$.

Definition 11 (Affine Boundedness of Updatable Garbling)**.** For a function family $\mathcal{F}_{\mathsf{pp}} \subseteq \mathcal{X}^2 \to \mathcal{Y}$ (Definition 7), and input bounds $\mathcal{X}' \subseteq \mathcal{X}$, let $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definition 8). Let $\mathcal{M} \subseteq \mathcal{X}$, $\mathcal{Y}$ be a quotient group of $\mathcal{X}$, and choose a right inverse $\iota\colon \mathcal{Y} \to \mathcal{X}$ of the projection from $\mathcal{X}$ to $\mathcal{Y}$. Let the set D of decodable inputs be the set of all $x \in \mathcal{X}$ such that $\iota(x) = x$. UGS satisfies affine boundedness if for all $\boldsymbol{x}_\sigma \in \mathcal{X}'$ and $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$ with $n(\rho_\sigma) \le T$ it holds that:

1. $\boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma) \in D$.
2. $\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma) \in D$.

and if for *all* $\rho_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$ (not necessarily of bounded norm),

1. $\Delta \in \mathsf{R}_{\mathsf{pk}} \mapsto \mathsf{MAC}(\rho_\sigma \star \Delta, \overline{\boldsymbol{x}}_\sigma)$ is an affine function.
2. The affine function $\mathsf{MAC}(\rho_\sigma, \cdot)$ has linear coefficient bounded by $\beta_{\mathsf{MAC}} \cdot n(\rho_\sigma)$ for some constant $\beta_{\mathsf{MAC}} \ge 1$.

Realizing updatable garbling from SIEDL groups. We instantiate updatable garbling schemes using SIEDL groups (Definition 5) and we do so in three stages. First, in Algorithm 1 we introduce the group R of ciphertext transformations used for rerandomizing, with its complete structure. Then in Algorithm 2 we define the samplers $\mathcal{G}$ and $\mathsf{Gen}_{\mathsf{mask}}$ by describing all of their subroutines. These differ only in that the elements sampled in $\mathsf{Gen}_{\mathsf{mask}}$ are from a larger space than those sampled in $\mathcal{G}$. Finally, in Algorithm 3 we define the various algorithms in UGS along with the default elements pk^* and pk_w.

Theorem 12. *For any SIEDL group where $q > B\ell$, Algorithms 1 to 3 describe an updatable garbling scheme (Definition 8) for bilinear functions $\mathcal{F}_{\mathsf{pp}}$, satisfying KDM security (Definition 10) for the set $\mathcal{F}^*_{\mathsf{pp}}$ of affine functions of the key into $\mathcal{F}_{\mathsf{pp}}$ (i.e., trilinear functions of the key and the gate inputs), and affine boundedness (Definition 11). It supports rerandomization up to G-seminorm $T = 2^{-\kappa-3}\sqrt{\frac{q}{B\ell}}$.*

The proof for this can be found in Appendix D.1.

3.2 Constant-Rate Rerandomizable Garbling Scheme

Using the updatable garbling scheme defined and instantiated in Section 3.1, in this section we construct a rerandomizable garbling scheme for arithmetic circuits for B-bounded computation over addition and multiplication gates. Our construction supports free-addition and requires two ciphertexts (containing 8 group elements each when instantiated with Theorem 12) to represent each multiplication gate. This scheme is detailed in Algorithms 4 to 6.

In our garbling scheme (1) addition of two wire values correspond to adding the corresponding wire labels; and (2) multiplication of two wire values involves using the wire labels to decrypt an encrypted multiplication gate (created using the updatable garbling scheme) to get the output wire label for that gate. Such a circuit also supports multiplication of a wire value with a public constant, simply by performing repeated addition of the corresponding wire label to itself.

Unfortunately, repeated addition causes the wire masks to grow exponentially, which affects the size of the ciphertext rerandomization factors ($\rho_{c_1,g}$ and $\rho_{c_2,g}$). Our scheme requires a bound on the size of the free additions, to know how big the ciphertext rerandomization masks generated in $\mathsf{Gen}_{\mathsf{mask}}$ need to be. This is formalized in the definition below.

Definition 13. The free-addition norm α of an arithmetic circuit is defined as follows. Every wire can be written as a linear function of input wires and outputs from multiplication gates; let $\boldsymbol{v}_\sigma$ be the coefficient vector of this linear function for wire σ. The free-addition norm of the circuit is the maximum of $\|v_\sigma\|_1$ over all input wires to multiplication gates.

Algorithm 1 Randomness Group for Updatable Garbling for Bilinear Functions

The ciphertext rerandomization group is split into rerandomization for each of two inputs. We start by defining a group R_{in} for rerandomizing each input:

$$\mathsf{R}_{\mathsf{in}} = \left\{ \begin{bmatrix} 1 & -m & -K & u_1 \\ 0 & 1 & -\Delta & u_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| (m, K, \Delta, u_1, u_2) \in \mathbb{Z}^5 \right\},$$

with matrix multiplication as $\star$. m shifts the input value, K shifts the input MAC, Δ shifts the global public key, and u_1, u_2 are encryption randomness. The G-seminorm is:

$$n(m, K, \Delta, u_1, u_2) = \max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2K + \Delta m|}{2^\kappa B \ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{|2u_1 + u_2 m|}{(2^\kappa B + 2)\ell}} \right).$$

We now describe its subgroups:

- $\overline{\mathsf{R}}_{\mathsf{in}}$ (isomorphic to $\langle \mathbb{Z}^2, + \rangle$) is the normal subgroup of only encryption randomness.
- $\mathsf{R}_{\mathsf{MAC}}$ (isomorphic to the Heisenberg group) is the wire rerandomization subgroup.
- $\overline{\mathsf{R}}_{\mathsf{MAC}}$ (isomorphic to $\langle \mathbb{Z}^2, + \rangle$) is the normal subgroup of $\mathsf{R}_{\mathsf{MAC}}$ that only shifts the input value and MAC.
- R_{pk} (isomorphic to $\langle \mathbb{Z}, + \rangle$) is the subgroup transforming only the global key pk.
- Every $\boldsymbol{R}_{\mathsf{in}} \in \mathsf{R}_{\mathsf{in}}$ can be written uniquely as $\overline{\boldsymbol{R}}_{\mathsf{in}} \star \overline{\boldsymbol{R}}_{\mathsf{MAC}} \star \boldsymbol{R}$, for $\overline{\boldsymbol{R}}_{\mathsf{in}} \in \overline{\mathsf{R}}_{\mathsf{in}}$, $\overline{\boldsymbol{R}}_{\mathsf{MAC}} \in \overline{\mathsf{R}}_{\mathsf{MAC}}$, and $\boldsymbol{R} \in \mathsf{R}_{\mathsf{pk}}$: extract (u_1, u_2) from $\boldsymbol{R}_{\mathsf{in}}$ to get $\overline{\boldsymbol{R}}_{\mathsf{in}}$, extract the global key Δ from $\boldsymbol{R}_{\mathsf{in}}$ to get $\boldsymbol{R}$, and finally extract (m, K) from $\boldsymbol{R}_{\mathsf{in}}$ and set $\overline{\boldsymbol{R}}_{\mathsf{MAC}}$ using $(m', K') = (m, K + \Delta m)$.

$$\mathsf{R}_{\mathsf{in}} = \overline{\mathsf{R}}_{\mathsf{in}} \star \mathsf{R}_{\mathsf{MAC}} \qquad \mathsf{R}_{\mathsf{MAC}} = \overline{\mathsf{R}}_{\mathsf{MAC}} \star \mathsf{R}_{\mathsf{pk}}$$

$$\overline{\mathsf{R}}_{\mathsf{in}} = \left\{ \begin{bmatrix} 1 & 0 & 0 & u_1 \\ 0 & 1 & 0 & u_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| (u_1, u_2) \in \mathbb{Z}^2 \right\} \qquad \overline{\mathsf{R}}_{\mathsf{MAC}} = \left\{ \begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| (m, K) \in \mathbb{Z}^2 \right\}$$

$$\mathsf{R}_{\mathsf{MAC}} = \left\{ \begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| (m, K, \Delta) \in \mathbb{Z}^3 \right\} \qquad \mathsf{R}_{\mathsf{pk}} = \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \Delta \in \mathbb{Z} \right\}$$

The ciphertext group R is the two copies of R_{in}, restricted to having the same global key Δ.

$$\mathsf{R} = \mathsf{R}_{\mathsf{in}} \times_{\mathsf{R}_{\mathsf{pk}}} \mathsf{R}_{\mathsf{in}} = \left\{ (\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2}) \,\middle|\, (\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2}) \in \mathsf{R}_{\mathsf{in}}{}^2, \mathsf{key}(\boldsymbol{R}_{\mathsf{in}\text{-}1}) = \mathsf{key}(\boldsymbol{R}_{\mathsf{in}\text{-}2}) \right\}.$$

It's G-seminorm is defined by $n((\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2})) = \max(n(\boldsymbol{R}_{\mathsf{in}\text{-}1}), n(\boldsymbol{R}_{\mathsf{in}\text{-}2}))$. The properties of R required by Definition 8 follow from the corresponding properties of R_{in}. In particular, it has these subgroups:

$$\overline{\mathsf{R}}_{\mathsf{in}\text{-}1} = \overline{\mathsf{R}}_{\mathsf{in}} \times 1 \qquad \overline{\mathsf{R}}_{\mathsf{in}\text{-}2} = 1 \times \overline{\mathsf{R}}_{\mathsf{in}} \qquad \overline{\mathsf{R}} = \overline{\mathsf{R}}_{\mathsf{in}} \times \overline{\mathsf{R}}_{\mathsf{in}}$$

$$\overline{\mathsf{R}}_{\mathsf{MAC},1} = \overline{\mathsf{R}}_{\mathsf{MAC}} \times 1 \qquad \overline{\mathsf{R}}_{\mathsf{MAC},2} = 1 \times \overline{\mathsf{R}}_{\mathsf{MAC}} \qquad \mathsf{R}_{\mathsf{pk}} = \{(\boldsymbol{R}, \boldsymbol{R}) \mid \boldsymbol{R} \in \mathsf{R}_{\mathsf{pk}}\}$$

$$\mathsf{R}_{\mathsf{MAC},1} = \{(\boldsymbol{R}_1, \mathsf{key}(\boldsymbol{R}_1)) \mid \boldsymbol{R}_1 \in \mathsf{R}_{\mathsf{MAC}}\} \qquad \mathsf{R}_{\mathsf{MAC},2} = \{(\mathsf{key}(\boldsymbol{R}_2), \boldsymbol{R}_2) \mid \boldsymbol{R}_2 \in \mathsf{R}_{\mathsf{MAC}}\}$$

And elements of R can be decomposed uniquely as a product $\overline{\mathsf{R}} \star \overline{\mathsf{R}}_{\mathsf{MAC},1} \star \overline{\mathsf{R}}_{\mathsf{MAC},2} \star \mathsf{R}_{\mathsf{pk}}$.

Algorithm 2 Samplers for Updatable Garbling for Bilinear Functions

$$\mathcal{G}_{\mathsf{offset}}(1^\kappa) = \{m \mid m \leftarrow [2^\kappa B]\}$$

$$\mathcal{G}_{\mathsf{pk}}(1^\kappa) = \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \, \Delta \leftarrow [\ell] \right\}$$

$$\mathcal{G}_{\mathsf{MAC}}(1^\kappa, m) = \left\{ \begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \, K \leftarrow \left[2^\kappa \sqrt{qB\ell}\right] \right\}$$

$$\mathcal{G}_{\mathsf{Enc\text{-}in}}(1^\kappa) = \left\{ \begin{bmatrix} 1 & 0 & 0 & u_1 \\ 0 & 1 & 0 & u_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \, u_1, u_2 \leftarrow [\ell] \right\}$$

$$\mathcal{G}_{\mathsf{Enc}}(1^\kappa) = \left\{ (\overline{\boldsymbol{R}}_{\mathsf{in\text{-}1}}, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}2}}) \,\middle|\, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}1}}, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}2}} \leftarrow \mathcal{G}_{\mathsf{Enc\text{-}in}}(1^\kappa) \right\}$$

$$\mathcal{G}^*_{\mathsf{pk}}(1^\kappa, \beta) = \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \, \Delta \leftarrow [2^\kappa \beta \ell] \right\}$$

$$\mathcal{G}^*_{\mathsf{MAC}}(1^\kappa, \beta) = \left\{ \begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \begin{array}{l} m \leftarrow [2^{2\kappa} \beta B] \\ K \leftarrow \left[2^{2\kappa} \beta^2 B\ell + 2^{2\kappa+1} \beta \sqrt{qB\ell}\right] \end{array} \right\}$$

$$\mathcal{G}^*_{\mathsf{Enc\text{-}in}}(1^\kappa, \beta) = \left\{ \begin{bmatrix} 1 & 0 & 0 & u_1 \\ 0 & 1 & 0 & u_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \middle| \begin{array}{l} u_1 \leftarrow [2^{2\kappa} \beta^2 B\ell] \\ u_2 \leftarrow [2^\kappa \beta \ell] \end{array} \right\}$$

$$\mathcal{G}^*_{\mathsf{Enc}}(1^\kappa, \beta) = \left\{ (\overline{\boldsymbol{R}}_{\mathsf{in\text{-}1}}, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}2}}) \,\middle|\, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}1}}, \overline{\boldsymbol{R}}_{\mathsf{in\text{-}2}} \leftarrow \mathcal{G}^*_{\mathsf{Enc\text{-}in}}(1^\kappa) \right\}$$

Note that α in Definition 13 is public, as it is easily calculated from the circuit topology, which is public. This value is used in algorithm 6 in the specification of $\mathsf{Gen}_{\mathsf{mask}}$.

Theorem 14. *For* $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$*, where* $\mathcal{X}$ *and* $\mathcal{Y}$ *are abelian groups written additively, let* $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ *be space of bilinear functions (Definition 7), and* $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ *be a UGS (Definition 8) satisfying rerandomizability (Definition 8), affine boundedness (Definition 11), and KDM security (Definition 10) for* $\mathcal{F}^*_{\mathsf{pp}}$ *with affine functions of the key into* $\mathcal{F}_{\mathsf{pp}}$*. Then Algorithms 4 to 6 presents a projective* T*-rerandomizable garbling scheme (Definition 3) for function family* $\mathcal{F} = \{f_{\mathsf{pp}} : \mathcal{X}^n \to \mathcal{Y}\}$ *of* B*-bounded arithmetic circuits. Its rerandomization bound* T *matches that of the underlying UGS, and it has mask* G*-seminorm bound* $\beta'_{mask} = (2\beta^2_{mask} + 4\beta_{mask} + 5)\beta_{mask}$ *where* β_{mask} *is the bound for the underlying UGS scheme.*

The proof for this can be found in Appendix E.

Communication Complexity. We note that in our construction, each multiplication gate requires 17 group elements. This can be seen as follows: garbling a multiplication gate uses two ciphertexts $c_{1,g}, c_{2,g} \in \mathbb{G}^{4\times 4}$ (one for $f_\times$, one for f_{MAC}), each of the form $C = \left[\, [\overline{C},\ \mathsf{pk}_1],\ [\mathsf{pk}_2^\top,\ 0] \,\right]$ as in Algorithm 3, where $\mathsf{pk}_1, \mathsf{pk}_2 \in \mathbb{G}^3$ are already known from the gate's input wires. The only

Algorithm 3 Updatable Garbling for Bilinear Functions

1. The public parameters are generated as in Definition 7. The rerandomization groups are defined in Algorithm 1, and the rerandomization samplers are defined in Algorithm 2.

2. The public keys and ciphertexts have the following form:

$$\begin{aligned} \mathsf{pk} &\in \mathbb{G} \\ \mathsf{pk}_\sigma &= [\overline{\mathsf{pk}}_\sigma \ \mathsf{pk} \ -\mathsf{g}]^\mathsf{T} \in \mathbb{G}^3 \\ \boldsymbol{C} &= \begin{bmatrix} \overline{\boldsymbol{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix} \in \mathbb{G}^{4\times 4}. \end{aligned}$$

   Addition for pk_σ is addition on the element $\overline{\mathsf{pk}}_\sigma \in \mathbb{G}$, and addition for $\boldsymbol{C}$ is addition on the 3×3 submatrix $\overline{\boldsymbol{C}}$.

3. Default elements:

$$\begin{aligned} \mathsf{pk}^* &= 0 \in \mathbb{G} \\ \mathsf{pk}_w &= [0 \ \ 0 \ \ -\mathsf{g}]^\mathsf{T} \in \mathbb{G}^3 \\ \mathsf{Compile}(\boldsymbol{F}) &= \begin{bmatrix} \boldsymbol{F}\cdot\mathsf{f} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0}^\mathsf{T} & 0 & -\mathsf{g} \\ \boldsymbol{0}^\mathsf{T} & -\mathsf{g} & 0 \end{bmatrix} \in \mathbb{G}^{4\times 4} \end{aligned}$$

4. Evaluation:

$$\mathsf{Eval}(\mathbf{C}, \boldsymbol{X}, \overline{\boldsymbol{x}}) = \mathsf{DLog}\left(\begin{bmatrix} 1 & \overline{\boldsymbol{x}}_0 & \boldsymbol{X}_0 & 0 \end{bmatrix} \cdot \mathbf{C} \cdot \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_1 \\ \boldsymbol{X}_1 \\ 0 \end{bmatrix} \right)$$

5. Rerandomization group action:

$$\begin{aligned} \mathsf{Re}_{\mathsf{pk}}(\Delta, \mathsf{pk}) &= \mathsf{pk} + \Delta \cdot \mathsf{g} \\ \begin{bmatrix} \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\boldsymbol{R}_\sigma, \mathsf{pk}_\sigma) \\ 0 \end{bmatrix} &= \boldsymbol{R}_\sigma \cdot \begin{bmatrix} \mathsf{pk}_\sigma \\ 0 \end{bmatrix} \\ \mathsf{Rerand}(\rho_c = (\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2}), c = \mathbf{C}) &= \boldsymbol{R}_{\mathsf{in}\text{-}1} \cdot \mathbf{C} \cdot \boldsymbol{R}_{\mathsf{in}\text{-}2}{}^\mathsf{T} \end{aligned}$$

6. Input encoding:

$$\begin{bmatrix} 1 \\ \mathsf{Re}_{\mathsf{MAC}}(\boldsymbol{R}_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)) \\ 0 \end{bmatrix} = \boldsymbol{R}_\sigma^{-\mathsf{T}} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma \\ \boldsymbol{X}_\sigma \\ 0 \end{bmatrix}$$

$$\mathsf{MAC}(\boldsymbol{R}_\sigma, \boldsymbol{x}_\sigma) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^\mathsf{T} \boldsymbol{R}_\sigma^{-\mathsf{T}} \begin{bmatrix} 1 \\ \boldsymbol{x}_\sigma \\ 0 \\ 0 \end{bmatrix} = K_\sigma + \Delta \cdot (\boldsymbol{x}_\sigma + m_\sigma)$$

7. Valid wires:

$$\mathcal{W} = \left\{ (\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \;\middle|\; [1 \ \overline{\boldsymbol{x}}_\sigma \ \boldsymbol{X}_\sigma] \cdot \mathsf{pk}_\sigma = 0 \right\}$$

8. Output decoding: $\iota(x) = x \bmod q \in [-q/2, q/2)$.

Algorithm 4 Rerandomizable Garbling for General Function Families

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8, 10 and 11). Then $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ is defined as:

1. $\mathsf{Gb}(f, \mathsf{pp}) \to (\boldsymbol{F}, \boldsymbol{d}, \boldsymbol{e})$
   - let $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$ be a circuit representing f
   - compute $(\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}}) = \mathsf{Compile}(\mathbf{C})$
   - sample $\rho \leftarrow \mathsf{Gen}(1^\kappa, \mathbf{C})$
   - compute $(\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}') = \mathsf{Upd}(\rho, (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}}))$
   - return $(\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}')$

2. $\mathsf{Compile}(\mathbf{C}) = (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}})$
   - parse $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$
   - let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be default public values for UGS
   - for all wires $\sigma \in [\mathsf{n} + \mathsf{Q}]$, set $\mathsf{pk}_\sigma = \mathsf{pk}_w$ and $\boldsymbol{m}_\sigma = 0$
   - for all gates $g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$ in topological order,
     - let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
     - set $f_\times \in \mathcal{F}_{\mathsf{pp}}$ s.t. $f_\times(x_1, x_2) = x_1 \times x_2$
     - set $f_{\mathsf{MAC}} \in \mathcal{F}_{\mathsf{pp}}$ s.t. $f_{\mathsf{MAC}}(x_1, x_2) = \mathsf{UGS.MAC}(1, f_\times(x_1, x_2))$
     - compute $c_{1,g} \leftarrow \mathsf{UGS.Compile}(f_\times)$ and $c_{2,g} \leftarrow \mathsf{UGS.Compile}(f_{\mathsf{MAC}})$
   - output $\begin{cases} \boldsymbol{F} & = (\mathsf{pk}^*, \{(c_{1,g}, c_{2,g}, \mathsf{pk}_w)\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\mathsf{pk}_w\}_{\sigma \in [\mathsf{n}]}) \\ \boldsymbol{d} & = \{0\}_{\sigma \in [\mathsf{m}]} \\ \boldsymbol{e} & = \{1\}_{\sigma \in [\mathsf{n}]} \end{cases}$

3. $\mathsf{Gen}(1^\kappa, \mathbf{C}) \to \rho$
   - parse $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$
   - sample $\Delta \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{pk}}(1^\kappa)$
   - for all wires $\sigma \in [\mathsf{n}]$, sample $\boldsymbol{m}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$ and $\overline{\rho}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{MAC}}(\boldsymbol{m}_\sigma)$
   - for all gates $g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$ in topological order, sample $\overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g} \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{Enc}}(1^\kappa)$ and for output wire σ, sample $\boldsymbol{m}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$ and $\overline{\rho}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{MAC}}(\boldsymbol{m}_\sigma)$
   - output $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}_\sigma\}_{\sigma \in [\mathsf{n}]})$

4. $\mathsf{Upd}(\rho_{\mathbf{C}}, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = (\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}')$
   - compute $(\boldsymbol{F}', \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) = \mathsf{Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F})$
   - output $\begin{cases} \boldsymbol{F}' = \boldsymbol{F}' \\ \boldsymbol{e}' = \pi_{\mathsf{in}}(\boldsymbol{e}) \\ \boldsymbol{d}' = \pi_{\mathsf{out}}(\boldsymbol{d}) \end{cases}$

new content of each ciphertext is therefore the 3×3 block $\overline{C}$; and since row and column 3 of every R_{in}-element equal $(0, 0, 1, 0)$, the $(3, 3)$ entry of C is invariant under Rerand and always equals 0 (its value in $\mathsf{Compile}(F)$), leaving only 8 elements per ciphertext. Together with the freshly-derived output-wire key $\mathsf{pk}_\sigma \in \mathbb{G}^3$ (note only the first coordinate is new, while the other two are shared across all wires) a multiplication gate costs $2 \times 8 + 1 = 17$ group elements.

Algorithm 5 Rerandomizable Garbling for General Function Families - contd.

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8, 10 and 11). Then $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ is defined as:

1. $\mathsf{Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F}) = (\boldsymbol{F'}, \pi_{\mathsf{out}}, \pi_{\mathsf{in}})$
   - parse $\rho_{\mathbf{C}} = (\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - parse $\boldsymbol{F} = (\mathsf{pk}, \{(c_{1,g}, c_{2,g}, \mathsf{pk}_\sigma)\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\mathsf{pk}_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - compute $\mathsf{pk}' = \mathsf{UGS.Re}_{\mathsf{pk}}(\Delta, \mathsf{pk})$
   - for all wires $\sigma \in [\mathsf{n}]$, compute $\mathsf{pk}'_\sigma = \mathsf{UGS.Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \Delta, \mathsf{pk}_\sigma)$
   - for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n}+\mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
     if $\mathsf{op}_g = +$,
     - set $\mathsf{pk}'_\sigma = \mathsf{pk}'_{\sigma_1} + \mathsf{pk}'_{\sigma_2}$, and $\overline{\rho}_\sigma = \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2}$

     if $\mathsf{op}_g = \times$,
     - compute $\mathsf{pk}'_\sigma = \mathsf{UGS.Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \Delta, \mathsf{pk}_\sigma)$
     - let $f_{\mathsf{offset}}(x) = x + \mathsf{offset}(\overline{\rho}_\sigma)$ be an affine function in x
     - set $c'_{1,g} = f_{\mathsf{offset}}(c_{1,g})$
     - let $f_{\mathsf{MAC}}(x) = \mathsf{UGS.MAC}(\overline{\rho}_\sigma \star \Delta, x)$ be an affine function in x
     - set $c'_{2,g} = c_{2,g} + f_{\mathsf{MAC}}(c_{1,g})$
     - for $i \in [2]$, set $c''_{i,g} = \mathsf{UGS.Rerand}(\overline{\rho}_{c_i,g} \star \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta, c'_{i,g})$
   - output $\begin{cases} \boldsymbol{F'} & = (\mathsf{pk}', \{(c''_{1,g}, c''_{2,g}, \mathsf{pk}'_\sigma)\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\mathsf{pk}'_\sigma\}_{\sigma\in[\mathsf{n}]}) \\ \pi_{\mathsf{out}} & = \{\boldsymbol{m}_{\mathsf{n}+\mathsf{Q}-\sigma}\}_{\sigma\in[\mathsf{m}]} \\ \pi_{\mathsf{in}} & = \{\rho_\sigma\}_{\sigma\in[\mathsf{n}]} \end{cases}$
2. $\pi_{\mathsf{out}}(\boldsymbol{d}) = \boldsymbol{d'}$
   - parse $\pi_{\mathsf{out}} = \{\boldsymbol{m'_\sigma}\}_{\sigma\in[\mathsf{m}]}$ and $\boldsymbol{d} = \{\boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$
   - output $\boldsymbol{d'} = \{\boldsymbol{m'_\sigma} + \boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$
3. $\pi_{\mathsf{in}}(\boldsymbol{e}) = \boldsymbol{e'}$
   - parse $\pi_{\mathsf{in}} = \{\rho'_\sigma\}_{\sigma\in[\mathsf{n}]}$ and $\boldsymbol{e} = \{\rho_\sigma\}_{\sigma\in[\mathsf{n}]}$
   - output $\boldsymbol{e'} = \{\rho_\sigma \star \rho'_\sigma\}_{\sigma\in[\mathsf{n}]}$
4. $\mathsf{En}(\boldsymbol{e}, \boldsymbol{x}) = \boldsymbol{X}$
   - parse $\boldsymbol{e} = \{\rho_\sigma\}_{\sigma\in[\mathsf{n}]}$
   - for each $\sigma \in [\mathsf{n}]$, set $\boldsymbol{X}_\sigma = \mathsf{UGS.MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)$
   - output $\boldsymbol{X} = \{(\boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma), \boldsymbol{X}_\sigma)\}_{\sigma\in[\mathsf{n}]}$
5. $\mathsf{De}(\boldsymbol{d}, \boldsymbol{Y}) = \boldsymbol{y}$
   - parse $\boldsymbol{d} = \{\boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$ and $\boldsymbol{Y} = \{(\boldsymbol{y'_\sigma}, \boldsymbol{Y}_\sigma)\}_{\sigma\in[\mathsf{m}]}$
   - for each $\sigma \in [\mathsf{m}]$, set $\boldsymbol{y}_\sigma = \boldsymbol{y'_\sigma} - \boldsymbol{m}_\sigma$
   - output $\boldsymbol{y} = \{\boldsymbol{y}_\sigma\}_{\sigma\in[\mathsf{m}]}$

Acknowledgments

This research was supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 101124977

Algorithm 6 Rerandomizable Garbling for General Function Families - contd.

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8, 10 and 11). Then $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ is defined as:

1. $\mathsf{Ev}(\boldsymbol{F}, \boldsymbol{X}) = \boldsymbol{Y}$
   - parse $\boldsymbol{F} = (\mathsf{pk}, \{(c_{1,g}, c_{2,g}, \mathsf{pk}_\sigma)\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\mathsf{pk}_\sigma\}_{\sigma\in[\mathsf{n}]})$ and $\boldsymbol{X} = \{(\boldsymbol{y}'_\sigma, \boldsymbol{Y}_\sigma)\}_{\sigma\in[\mathsf{n}]}$
   - for each $g \in [\mathsf{Q}]$, gate op_g in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n}+\mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
     - if $\mathsf{op}_g = +$, set $\boldsymbol{y}'_\sigma = \boldsymbol{y}'_{\sigma_1} + \boldsymbol{y}'_{\sigma_2}$, and $\boldsymbol{Y}_\sigma = \boldsymbol{Y}_{\sigma_1} + \boldsymbol{Y}_{\sigma_2}$
     - if $\mathsf{op}_g = \times$,
       * compute $\boldsymbol{y}'_\sigma = \iota(\mathsf{UGS.Eval}(c_{1,g}, \{\boldsymbol{Y}_{\sigma_i}\}_{i\in\{1,2\}}, \{\boldsymbol{y}'_{\sigma_i}\}_{i\in\{1,2\}}))$
       * compute $\boldsymbol{Y}_\sigma = \iota(\mathsf{UGS.Eval}(c_{2,g}, \{\boldsymbol{Y}_{\sigma_i}\}_{i\in\{1,2\}}, \{\boldsymbol{y}'_{\sigma_i}\}_{i\in\{1,2\}}))$
   - output $\boldsymbol{Y} = \{(\boldsymbol{y}'_{\mathsf{n}+\mathsf{Q}-\sigma}, \boldsymbol{Y}_{\mathsf{n}+\mathsf{Q}-\sigma})\}_{\sigma\in[\mathsf{m}]}$

2. $\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C}, \beta) \to \rho$
   - parse $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g\in[\mathsf{Q}]})$
   - sample $\Delta \leftarrow \mathsf{UGS}.\mathcal{G}^*_{\mathsf{pk}}(1^\kappa, \beta)$
   - for all wires $\sigma \in [\mathsf{n}]$, sample $\overline{\rho}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}^*_{\mathsf{MAC}}(1^\kappa, 2(\beta_{\mathrm{mask}}+1)\beta)$
   - let α be the free-addition norm Definition 13 of $\mathbf{C}$.
   - for all gates $g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$ in topological order, sample $\overline{\rho}_{c_1,g} \leftarrow \mathsf{UGS}.\mathcal{G}^*_{\mathsf{Enc}}(1^\kappa, (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}} + 8\alpha + 6)\beta)$, $\overline{\rho}_{c_2,g} \leftarrow \mathsf{UGS}.\mathcal{G}^*_{\mathsf{Enc}}(1^\kappa, (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}})\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta^2)$, and for the output wire σ, sample $\overline{\rho}_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}^*_{\mathsf{MAC}}(1^\kappa, 2(\beta_{\mathrm{mask}}+1)\beta)$
   - output $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}_\sigma\}_{\sigma\in[\mathsf{n}]})$

3. For $\rho^2, \rho^1 \in \mathsf{R}_{\mathsf{re}}$, the group operator $\rho^2 \star \rho^1 = \rho$ works as follows:
   - parse $\rho^1 = (\Delta^1, \{\overline{\rho}^1_\sigma, \overline{\rho}^1_{c_1,g}, \overline{\rho}^1_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}^1_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - parse $\rho^2 = (\Delta^2, \{\overline{\rho}^2_\sigma, \overline{\rho}^2_{c_1,g}, \overline{\rho}^2_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}^2_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - set $\Delta' = \Delta^2 \star \Delta^1$
   - for all $\sigma \in [\mathsf{n}]$, for $\rho^2_\sigma = \overline{\rho}^2_\sigma \star \Delta^2$ and $\rho^1_\sigma = \overline{\rho}^1_\sigma \star \Delta^1$, let $\overline{\rho}'_\sigma \star \Delta' = \rho^2_\sigma \star \rho^1_\sigma$
   - for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n}+\mathsf{Q}]$ be the indices of the input wires and output wire to op_g respectively
     if $\mathsf{op}_g = +$, set $\overline{\rho}'_\sigma = \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2}$
     if $\mathsf{op}_g = \times$,
     - for $\rho^2_\sigma = \overline{\rho}^2_\sigma \star \Delta^2$ and $\rho^1_\sigma = \overline{\rho}^1_\sigma \star \Delta^1$, let $\overline{\rho}'_\sigma \star \Delta' = \rho^2_\sigma \star \rho^1_\sigma$
     - for $i \in [2]$, let $\rho^2_{c_i,g} = \overline{\rho}^2_{c_i,g} \star \overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2$ and $\rho^1_{c_i,g} = \overline{\rho}^1_{c_i,g} \star \overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1$
     - compute $a_2 = \mathsf{UGS.MAC}(\rho^2_\sigma, 1) - \mathsf{UGS.MAC}(\rho^2_\sigma, 0)$
     - set $\overline{\rho}'_{c_1,g} \star \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \star \Delta' = \rho^2_{c_1,g} \star \rho^1_{c_1,g}$
     - set $\overline{\rho}'_{c_2,g} \star \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \star \Delta' = \rho^2_{c_2,g} \star (\overline{\rho}^1_{c_1,g})^{a_2} \star \rho^1_{c_2,g}$
       $\overline{\rho}'_{\sigma_1}, \overline{\rho}'_{\sigma_2}, \Delta'$ are consistent across equations, as $\mathsf{in}_1, \mathsf{in}_2, \mathsf{key}$ are homomorphisms
   - output $\rho = (\Delta', \{\overline{\rho}'_\sigma, \overline{\rho}'_{c_1,g}, \overline{\rho}'_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}'_\sigma\}_{\sigma\in[\mathsf{n}]})$

(DECRYPSYS). L. Roy was also supported the Swiss National Science Foundation (SNSF) under the Starting grant "Low-Entropy Cryptography for the Quantum Era".

References

[ADOS22] Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl. An algebraic framework for silent preprocessing with trustless setup and active security. In *CRYPTO 2022*, pages 421–452, 2022.

[AHKP22] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. SCALES - MPC with small clients and larger ephemeral servers. In *TCC 2022*, pages 502–531, 2022.

[AHKP24] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. Malicious security for SCALES - outsourced computation with ephemeral servers. In *CRYPTO 2024*, pages 3–38, 2024.

[AHKP25] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. Towards building efficient SCALES protocols. In *ASIACRYPT 2025*, pages 35–67, 2025.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 120–129. IEEE Computer Society, 2011.

[Ben87] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO 2010*, pages 1–20, 2010.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016*, pages 509–539, 2016.

[BGI+18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *ITCS 2018*, pages 21:1–21:21, 2018.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2008.

[BHJL17] Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. *J. Cryptol.*, pages 519–549, 2017.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, pages 784–796, 2012.

[BLLL23] Marshall Ball, Hanjun Li, Huijia Lin, and Tianren Liu. New ways to garble arithmetic circuits. In *EUROCRYPT 2023*, pages 3–34, 2023.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *CRYPTO '98*, pages 424–441, 1998.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *ACM SIGSAC CCS 2014*, pages 597–608, 2014.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In *CT-RSA 2015*, pages 487–505, 2015.

[CL24] Yi-Hsiu Chen and Yehuda Lindell. Optimizing and implementing fischlin's transform for uc-secure zero knowledge. *IACR Commun. Cryptol.*, 1(2):11, 2024.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC 2001*, pages 119–136, 2001.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO 2000*, pages 432–450, 2000.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, pages 152–168, 2005.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In *CRYPTO*, pages 155–172, 2010.

[HvdHJ25] Raphael Heitjohann, Jonas von der Heyden, and Tibor Jager. Rerandomizable garbling, revisited. In *CRYPTO 2025*, pages 619–651, 2025.

[HY24] Carmit Hazay and Yibin Yang. Toward malicious constant-rate 2pc via arithmetic garbling. In *EUROCRYPT 2024*, pages 401–431, 2024.

[ILL24] Yuval Ishai, Hanjun Li, and Huijia Lin. Succinct partial garbling from groups and applications. *IACR Cryptol. ePrint Arch.*, page 2073, 2024.

[ILL25] Yuval Ishai, Hanjun Li, and Huijia Lin. A unified framework for succinct garbling from homomorphic secret sharing. In *CRYPTO 2025*, pages 390–425, 2025.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *STOC*, page 20–31, 1988.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.

[LR25] Anna Lysyanskaya and Leah Namisa Rosenbloom. Adaptive UC NIZK for practical applications. In *LATINCRYPT 2025*, volume 16129, pages 76–108, 2025.

[LWYY25] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. Bitgc: Garbled circuits with 1 bit per gate. In *EUROCRYPT 2025*, pages 437–466, 2025.

[MORS24] Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Rate-1 arithmetic garbling from homomorphic secret sharing. In *TCC 2024*, pages 71–97, 2024.

[MORS25a] Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Computationally succinct authentication from DCR: attribute-based laconic function evaluation and more. *IACR Cryptol. ePrint Arch.*, page 2307, 2025.

[MORS25b] Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Silent circuit relinearisation: Sublinear-size (boolean and arithmetic) garbled circuits from DCR. In *CRYPTO 2025*, pages 426–458, 2025.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In *EUROCRYPT 2015*, pages 657–686, 2015.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *CCS '98*, pages 59–66, 1998.

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021*, pages 678–708, 2021.

[RS21] Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO 2021*, pages 687–717, 2021.

[Sch97] Eric Schechter. Chapter 26 - metrization of groups and vector spaces. In *Handbook of Analysis and Its Foundations*, pages 688–720. Academic Press, San Diego, 1997.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FoCS*, pages 162–167, 1986.

AI Declaration.

AI was used to standardize notation in the technical sections in response to reviewer feedback and to improve the wording and presentation of the technical overview and introduction. It was not used to generate technical ideas or results.

Supplementary Material

A Additional Preliminaries

Definition 15. Two ensembles of probability distributions $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are *computationally indistinguishable*, denoted $X \overset{\text{c}}{\approx} Y$, if for every PPT distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $\kappa \in \mathbb{N}$,

$$\left| \Pr_{x\leftarrow X_\kappa}[D(1^\kappa, x) = 1] - \Pr_{y\leftarrow Y_\kappa}[D(1^\kappa, y) = 1] \right| < \epsilon(\kappa).$$

Definition 16 (Invertible sampling [DN00])**.** A PPT sampler $\mathcal{G}(1^\kappa)$ has invertible sampling if there a PPT reverse-sampler Sim such that

$$\boxed{\begin{array}{l}\mathsf{coins} \leftarrow \{0,1\}^{\mathbb{N}} \\ x = \mathcal{G}(\mathsf{coins}; 1^\kappa) \\ \text{return } (x, \mathsf{coins})\end{array}} \overset{\text{s}}{\approx} \boxed{\begin{array}{l} x \leftarrow \mathcal{G}(1^\kappa) \\ \mathsf{coins} \leftarrow \mathsf{Sim}(x) \\ \text{return } (x, \mathsf{coins})\end{array}}.$$

Global Random Oracle Model. The global random oracle model of [CJS14] enables construction of UC NIZK proofs. In this model the random oracle is not programable, but it is observable in the sense that the simulator for a given label are given all queries to the random oracles pertaining to the label that is being simulated.

Global Random Oracle $\mathcal{G}_{\mathsf{gRO}}$

Parameters: output length $\ell(n)$ and a list $\bar{\mathcal{F}}$ of ideal functionality programs.

1. Upon receiving a query x, from some party $P = (\mathsf{pid}, \mathsf{sid})$ or from the adversary S do:
   - If there is a pair (x, v) for some $v \in \{0,1\}^{\ell(n)}$ in the (initially empty) list $\mathcal{Q}$ of past queries, return v to P. Else, choose uniformly $v \in \{0,1\}^{\ell(n)}$ and store the pair (x, v) in $\mathcal{Q}$. Return v to P.
   - Parse x as (s, x'). If $\mathsf{sid} \neq s$ then add (s, x', v) to the (initially empty) list of $\mathsf{illegitimate}$ queries for SID s, that we denote by $\mathcal{Q}_{|s}$.
2. Upon receiving a request from an instance of an ideal functionality in the list $\bar{\mathcal{F}}$, with SID s, return to this instance the list $\mathcal{Q}_{|s}$ of illegitimate queries for SID s.

Figure 7 Global Random Oracle [CJS14]

In the following we phrase the constructions in a slightly simplified model, which is equivalent. We assume a labelled random oracle where all queries belong to some label. These labels are used to segment the input space.

Definition 17 (Labelled random oracle)**.** A labelled random oracle defines a function $\mathsf{gRO} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^\kappa$, where κ is the security parameter, via lazy evaluation as follows:

- maintain an initially empty map $\mathcal{Q}$ of past queries.
- on input $(L, x) \in \{0,1\}^* \times \{0,1\}^*$,
  - if there is some $((L, x), v) \in \mathcal{Q}$ then return v.
  - otherwise, sample $v \leftarrow \{0,1\}^\kappa$, add $((L, x), v)$ to $\mathcal{Q}$, and return v.

For all *labels* $L \in \{0,1\}^*$ let $\mathcal{Q}_{|L} = \{((L', x), v) \in \mathcal{Q} | L' = L\}$.

The idea behind [CJS14] is that all labels belong to a session. In particular, in UC these are classified as: (1) labels L belonging to honest parties in the challenge session (HC); (2) labels belonging to the corrupt parties in the challenge session (CC); and (3) non-challenge session labels (NC). In the framework, the simulator gets access to $\mathsf{gRO}_{|L}$ for all CC labels, allowing it to extract witnesses from proofs given by corrupted parties in the challenge session. The environment gets access to $\mathsf{gRO}_{|L}$ for all NC labels but not for any HC label. We refer the reader to [CJS14] for details.

A.1 Σ-Protocols and Zero-Knowledge Proofs

Zero-Knowledge Proofs are an interaction between a prover P and a verifier V such that the interaction convinces V that an instance x of P belongs to a public language L but the interaction reveals nothing beyond that fact. In particular, the interaction should satisfy correctness (correct statements can always be convincingly proven), and soundness (incorrect statements cannot be convincingly proven, except with negligible probability).

Definition 18. A pair of machines (P, V) is a **computational zero-knowledge** proof for a relation $R(x, w)$ over statements x and witnesses w if for every PPT V, the following holds:

- **Completeness:** For every $(x, w) \in R$,
$$\Pr[\langle P(w, x), V(x)\rangle = 1] > 1 - negl(\kappa)$$
- **ϵ-Soundness:** For every x for which $\nexists w$ s.t. $(x, w) \in R$, and every P^*,
$$\Pr[\langle P^*, V\rangle(x) = 1] \leq \epsilon$$
- **Zero-Knowledge:** For every PPT V^* there exists a PPT S^* such that,
$$\{\langle P(w, x), V^*(x)\rangle\}_{(x,w)\in R} \overset{c}{\approx} \{S^*(x)\}_{(x,w)\in R}$$
- **Knowledge Soundness:** There exists a PPT E that receives oracle access to P^*, s.t. for every x for which $\exists w$ s.t. $(x, w) \in R$, and every P^*,
$$\Pr[\langle P^*, V\rangle(x) = 1] - \Pr[E^{P^*(x)}(x) \to w' \wedge (x, w') \in R] \leq \epsilon$$

In this work, we present Σ-protocols [CDS94] that prove the correctness of garbling and correct rerandomizing. These can be converted into universally composable zero-knowledge proofs of knowledge [LR25, CL24] for the same statements where the prover sends only one message, using the Fischlin transform [Fis05].

Definition 19. A **Σ-protocol** $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ for a statement $x \in L$ with witness w is an interaction between a prover P and verifier V of the form:

1. P computes $\mathsf{aux}, a \leftarrow P_1(x, w)$ and sends the message a to V
2. V computes $b \leftarrow V_1(\cdot)$, a uniformly random b to send to P
3. P sends the last message $c = P_2(a, b, x, w)$ to V
4. V outputs $V_2(a, b, c, x) \in \{0, 1\}$

Such a protocol must satisfy the following properties:

- **Completeness**: For every $x \in L$, $a \leftarrow P_1(x, w)$, and $c = P_2(a, b, x, w)$,
$$\Pr[V_2(a, b, c, x) = 1] > 1 - negl(\kappa)$$

- **Special Soundness**: Let w be a witness proving $x \in L$, $a \leftarrow P_1(x, w)$. Then for any set of transcripts (a, b, c) and (a, b', c') s.t. $b \neq b'$, $V_2(a, b, c, x) = 1$ and $V_2(a, b', c', x) = 1$, there exists a PPT algorithm,
$$w \leftarrow \mathsf{Extract}(a, b, c, b', c', x)$$
- **Special Honest-Verifier Zero Knowledge (SHVZK)**: There exists a PPT algorithm,
$$\tau = (a, b, c) \leftarrow \mathsf{Sim}(1^\kappa, x, b)$$
such that $1 = V_2(a, b, c, x)$. For all $b \in \{0,1\}^\ell$, the distribution $\tau \leftarrow \mathsf{Sim}(1^\kappa, x, b)$ is computationally indistinguishable from the distribution of $\tau' = (a', b, c')$ obtained from an honest execution of $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ for a statement $x \in L$ with witness w.

The Fischlin transform [Fis05] is a standard compilation that converts any Σ-protocol (Definition 19), where (1) the verifier's challenge b belongs in a large space $\{0,1\}^{O(\kappa)}$, and (2) the (possibly cheating) prover's second message is, verifiably, a deterministic function of the transcript thus far, into a non-interactive zero-knowledge proof of knowledge with one proof string from the prover to the verifier. This transformation makes use of an observable random oracle $\mathsf{gRO} : A \to \{0,1\}^{O(\kappa)}$ (Definition 17 – appendix A), where A is the space of all prover's first messages, appended with a nonce. Letting $n \leftarrow \{0,1\}^{O(\kappa)}$ be a nonce, it works by having the prover send $\pi = (a, n, b, c)$ to the verifier where, $\mathsf{aux}, a \leftarrow P_1(x, w)$, $0^\ell || b \leftarrow \mathcal{R}(a||n)$, and $c = P_2(a, b, x, w)$. Informally, knowledge extraction works by observing the prover's queries to gRO for various n (until the output prefix is 0^ℓ) to output two different challenges for the same first message. The completeness, soundness and zero-knowledge properties follow from the properties of the Σ-protocol.

B Malicious Security for Rerandomizable Garbling

In this section, we present Σ-protocols proving the correctness of garbling and rerandomizing, that are black-box in the garbling scheme in Definition 3. This can be converted into a zero-knowledge proof of knowledge using transformations well-known from prior work. The high-level idea follows that in [CD98] which presents zero-knowledge proofs of knowledge of the underlying message in a commitment scheme when this message is an element of a finite field. We first present the proof for correct rerandomizing and then discuss how this can be extended to a proof of correct garbling.

B.1 Proof of Correct Rerandomizing

We present a Σ-protocol for the statement given in Equation (1) with soundness error $\frac{1}{2}$. This soundness can be amplified to give negligible soundness error using parallel repetition.

$$\exists \pi_{\mathsf{in}}, \rho_{\mathbf{C}} \in \mathsf{R_{re}}' \text{ s.t. } (\boldsymbol{F'}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) \leftarrow \mathsf{RGS.Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F}) \tag{1}$$

Our protocol, formalized in Algorithm 8, follows a similar outline as that of [AHKP24] and is black-box in the underlying RGS. It incurs a total communication complexity of approximately twice the size of the function garbling.

Theorem 20. *Assuming* $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is an RGS (Definition 3) for function family* $\mathcal{F}$*, the protocol in Algorithm 8 is a* Σ*-protocol (Definition 19) for the statement in Equation* (1) *with soundness error* $\frac{1}{2}$*.*

The complete proof for this can be found in Appendix F.

Algorithm 8 Σ-Protocol for Correct Rerandomizing (Equation (1))

For a function family $\mathcal{F}$, let $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ be a rerandomizable garbling scheme (Definition 3). Then $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ is defined as follows:

1. $P_1(x, w) \to (a, \mathsf{aux})$
   - parse $w = \rho_{\mathbf{C}}$ and $x = (\boldsymbol{F}, \boldsymbol{F'})$
   - sample $\rho^* \leftarrow \mathsf{RGS.Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C})$
   - compute $(\boldsymbol{F'}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) = \mathsf{RGS.Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F})$ and $(\boldsymbol{F}_{\mathsf{pf}}, \pi'_{\mathsf{in}}, \pi'_{\mathsf{out}}) = \mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F'})$
   - output $\begin{cases} a & = \boldsymbol{F}_{\mathsf{pf}} \\ \mathsf{aux} & = \rho^* \end{cases}$
2. $V_1 \to b$
   - output $b \leftarrow \{0, 1\}$
3. $P_2(b, x, w, \mathsf{aux}) = c$
   - parse $w = \rho_{\mathbf{C}}$ and $\mathsf{aux} = \rho^*$
   - output $c = \begin{cases} \rho^* & \text{if } b = 0 \\ \rho^* \star \rho_{\mathbf{C}} & \text{if } b = 1 \end{cases}$
4. $V_2(a, b, c, x) \in \{\bot, \top\}$
   - parse $c = \rho$ and if $\rho \notin \mathit{supp}(\mathsf{RGS.Gen}_{\mathsf{mask}})$, output $\bot$
   - parse $a = \boldsymbol{F}_{\mathsf{pf}}$
   - if $b = 0$, if $\mathsf{RGS.Rerand}(\rho, \boldsymbol{F'})_0 == \boldsymbol{F}_{\mathsf{pf}}$, output $\top$
   - if $b = 1$, if $\mathsf{RGS.Rerand}(\rho, \boldsymbol{F})_0 == \boldsymbol{F}_{\mathsf{pf}}$, output $\top$
   - otherwise, output $\bot$

B.2 Proof of Correct Garbling

Proving the correctness of garbling in an RGS corresponds to a proof for the following statement.

$$\exists \boldsymbol{e}, \rho_{\mathbf{C}} \in \mathsf{R_{re}}' \text{ s.t. } (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}) \leftarrow \mathsf{RGS.Gb}(\rho_{\mathbf{C}}, \mathbf{C}) \tag{2}$$

Owing to the properties of RGS (Definition 3), note that this statement is equivalent to,

$$\exists \pi_{\mathsf{in}}, \rho_{\mathbf{C}} \in \mathsf{R_{re}} \text{ s.t. } (\boldsymbol{F}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) \leftarrow \mathsf{RGS.Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F}_{\mathsf{pub}})$$

where $(\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}}) = \mathsf{Compile}(\mathbf{C})$ is something that the verifier can compute publicly, $\boldsymbol{e} = \pi_{\mathsf{in}}(\boldsymbol{e}_{\mathsf{pub}})$ and $\boldsymbol{d} = \pi_{\mathsf{out}}(\boldsymbol{d}_{\mathsf{pub}})$. It therefore follows that the protocol in Algorithm 8 is also a Σ-protocol for for the statement in Equation (2) with soundness error $\frac{1}{2}$.

Corollary 21. *Assuming* $\mathsf{RGS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is an RGS (Definition 3) for function family* $\mathcal{F}$*, the protocol in Algorithm 8 is a* Σ*-protocol (Definition 19) for the statement in Equation* (2) *with soundness error* $\frac{1}{2}$.

C Proof of Lemma 6

We prove each of the three statements in order.

Claim 1. $$\{\mathsf{pp}, \mathsf{h}\}_{\mathsf{h} \leftarrow \mathbb{H}} \overset{\mathrm{c}}{\approx} \{\mathsf{pp}, \mathsf{h} + y \cdot \mathsf{f}\}_{\mathsf{h} \leftarrow \mathbb{H}}$$

Proof. We give a hybrid argument, from left to right.

Hybrid_1: This is the left distribution:

$$\{\mathsf{pp}, \mathsf{h}\}_{\mathsf{h} \leftarrow \mathbb{H}}$$

Hybrid_2: Use Subgroup Indistinguishability to change h to instead be sampled as $\mathsf{h} \leftarrow \mathbb{G}$

Hybrid_3: Replace h with $\mathsf{h} + y \cdot \mathsf{f}$. This is statistically indistinguishable as both distributions are uniform on $\mathbb{G}$. That is, h acts as a one-time pad.

Hybrid_4: Use Subgroup Indistinguishability to change h back to being sampled as $\mathsf{h} \leftarrow \mathbb{H}$. We are now at the right distribution.

□

Claim 2. $\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [\ell]} \overset{c}{\approx} \{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h} + y \cdot \mathsf{f}\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [\ell]}$

Proof. We give a hybrid argument, from left to right.

Hybrid_1: Let L be an upper bound on the cardinality of $\mathbb{G}$; one can be found easily using the bit length of elements of $\mathbb{G}$. Instead of sampling $x \leftarrow [\ell]$, sample $x \leftarrow [2^\kappa L]$. Before this change, the ℓ-Bounded Keys property (Definition 5) shows that $x \cdot \mathsf{h}$ is indistinguishable from uniform in $\langle \mathsf{h} \rangle$. After this change, x is sampled from a wide enough range that the statistical distance of $x \cdot \mathsf{h}$ from uniform on $\langle \mathsf{h} \rangle$ is at most $\frac{|\langle \mathsf{h} \rangle|}{2^\kappa L} \leq 2^{-\kappa}$, which is negligible. Therefore, by the ℓ-Bounded Keys property (Definition 5), this change is indistinguishable.

Hybrid_2: Replace $\mathsf{h} \leftarrow \mathbb{H}$ with $\mathsf{h}' \leftarrow \mathbb{H}; \mathsf{h} = \mathsf{h}' + \mathsf{f}$. This is change is indistinguishable by Claim 1.

Hybrid_3: Since x is only used for multiplication by h, which has order dividing $|\mathbb{H}| \cdot q$, x only matters mod $|\mathbb{H}| \cdot q$. Let $\overline{x} = x \bmod |\mathbb{H}|$ and $\hat{x} = x \bmod q$. Since $|\mathbb{H}|$ and q are coprime, use CRT to define $x \bmod |\mathbb{H}| \cdot q$ in terms of $\overline{x} \leftarrow [|\mathbb{H}|]$ and $\hat{x} \in [q]$. This new distribution makes x exactly uniform mod $|\mathbb{H}| \cdot q$, while the previous hybrid had statistical distance at most $\frac{|\mathbb{H}| \cdot q}{2^\kappa L} \leq 2^{-\kappa}$ from uniform mod $|\mathbb{H}| \cdot q$. Therefore, this hybrid is statistically indistinguishable.

Note that the use of $|\mathbb{H}|$ means that the experiment is now inefficient. This means we must remove the use of $|\mathbb{H}|$ before using any computational security properties.

Hybrid_4: Write

$$x \cdot \mathsf{h} = x \cdot \mathsf{h}' + x \cdot \mathsf{f} = \overline{x} \cdot \mathsf{h}' + \hat{x} \cdot \mathsf{f},$$

since f has order q and h' has order dividing $|\mathbb{H}|$.

Hybrid_5: Shift $\hat{x}$ up by y. This is indistinguishable because $\hat{x}$ and $\hat{x} + y$ are both uniform mod q. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}' + \mathsf{f}, \overline{x} \cdot \mathsf{h}' + \hat{x} \cdot f + y \cdot f\}_{\mathsf{h}' \leftarrow \mathbb{H}, \overline{x} \leftarrow [|\mathbb{H}|], \hat{x} \leftarrow [q]}.$$

Hybrid_6: Undo Hybrid_4 to again write things in terms of the CRT x of $\overline{x}$ and $\hat{x}$, i.e., $x \cdot \mathsf{h} = \overline{x} \cdot \mathsf{h}' + \hat{x} \cdot \mathsf{f}$

Hybrid_7: Undo Hybrid_3. That is, sample x from a distribution $x \leftarrow [2^\kappa L]$ sufficiently wide that $x \cdot \mathsf{h}$ is statistically $2^{-\kappa}$-close to uniform, while in the previous hybrid it was exactly uniform. After this change, $|\mathbb{H}|$ is no longer used, so the experiment is now efficient again. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}' + \mathsf{f}, x \cdot (\mathsf{h}' + \mathsf{f}) + y \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L]}.$$

Hybrid_8: Undo Hybrid_2, i.e., switch from $\mathsf{h} = \mathsf{h}' + \mathsf{f}$ back to $\mathsf{h} \leftarrow \mathbb{H}$, using Claim 1. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h} + y \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L]}.$$

Hybrid_9: Undo Hybrid_1, i.e., switch the range for x back to $[\ell]$. The previous hybrid had $x \cdot \mathsf{h}$ statistically close to uniform, and the current hybrid is indistinguishable from uniform by ℓ-Bounded Keys. We are now at the right distribution.

□

Claim 3.

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h}, x' \cdot x \cdot \mathsf{h}\}_{\mathsf{h} \leftarrow \mathbb{H}, x, x' \leftarrow [\ell]} \overset{c}{\approx} \{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + y \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + y' \cdot \mathsf{f}\}_{\mathsf{h} \leftarrow \mathbb{H}, x, x' \leftarrow [\ell]}$$

Proof. We prove a version in which $y, y' \leftarrow \mathbb{Z}_q$ are uniformly random. A two-step hybrid argument then shows the original claim in which y and y' are chosen.

When now give a hybrid argument from the left distribution to random.

Hybrid_1: Let L be an upper bound on the cardinality of $\mathbb{G}$. Instead of sampling $x \leftarrow [\ell]$, sample $x \leftarrow [2^\kappa L]$. This is indistinguishable by the ℓ-Bounded Keys property (Definition 5).

Hybrid_2: Replace $x' \cdot \mathsf{h}$ with $x' \cdot \mathsf{h} + \mathsf{f}$. This is change is indistinguishable by Claim 2. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + \mathsf{f}, x' \cdot x \cdot \mathsf{h} + x \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell]}.$$

Hybrid_3: Sample $u \leftarrow \mathbb{Z}_q$, and replace x with $x + |\mathbb{H}| \cdot (u\,|\mathbb{H}|^{-1} \bmod q)$. This change shifts x by at most $|\mathbb{H}| \cdot q$, and so has statistical distance at most $\frac{|\mathbb{H}| \cdot q}{2^\kappa L} \leq 2^{-\kappa}$. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + \mathsf{f}, x' \cdot x \cdot \mathsf{h} + x \cdot f + u \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell], u \leftarrow \mathbb{Z}_q}.$$

Hybrid_4: Replace $x' \cdot \mathsf{h}$ with $x' \cdot \mathsf{h} + (y-1) \cdot \mathsf{f}$. This is change is indistinguishable by Claim 2. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + y \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + y \cdot x \cdot f + u \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell], y \leftarrow \mathbb{Z}_q, u \leftarrow \mathbb{Z}_q}.$$

Hybrid_5: Replace u with $y' - y \cdot x$, where $y' \leftarrow \mathbb{Z}_q$. This is indistinguishable because both values are uniform mod q. The distribution is now

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + y \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + y' \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell], y \leftarrow \mathbb{Z}_q, y' \leftarrow \mathbb{Z}_q}.$$

Hybrid_6: Undo Hybrid_1, i.e., use the ℓ-Bounded Keys property to sample $x \leftarrow [\ell]$ instead of $x \leftarrow [2^\kappa L]$. We are now at the random distribution.

To get the original claim, first use the hybrids above to change to

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + z \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + z' \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell], z \leftarrow \mathbb{Z}_q, z' \leftarrow \mathbb{Z}_q},$$

then shift z and z' by y and y', respectively, and finally change back from random to the right distribution by undoing the hybrids above, to get

$$\{\mathsf{pp}, \mathsf{h}, x \cdot \mathsf{h}, x' \cdot \mathsf{h} + y \cdot \mathsf{f}, x' \cdot x \cdot \mathsf{h} + y' \cdot f\}_{\mathsf{h} \leftarrow \mathbb{H}, x \leftarrow [2^\kappa L], x' \leftarrow [\ell]}$$

for chosen $y, y' \in \mathbb{Z}_q$. □

D Proofs for the Updatable Garbling Scheme

D.1 Proof of Theorem 12

Proof. We show that UGS in Algorithms 1 to 3 satisfies Definitions 8 and 10 as follows:

Group stucture. The groups described in Algorithm 1 are groups, as upper triangular matrices (with 1s on the diagonal) are closed under multiplication and have determinant 1, and by inspection matrix multiplication won't cause one of the 0 entries in these subgroups to become nonzero.

The decomposition into $\overline{\boldsymbol{R}}_{\mathsf{in}} \star \overline{\boldsymbol{R}}_{\mathsf{MAC}} \star \boldsymbol{R}$ is correct:

$$\begin{bmatrix}1&0&0&u_1\\0&1&0&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix} \star \begin{bmatrix}1&-m&-K-\Delta m&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \star \begin{bmatrix}1&0&0&0\\0&1&-\Delta&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&u_1\\0&1&0&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix} \star \begin{bmatrix}1&-m&-K&0\\0&1&-\Delta&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&-m&-K&u_1\\0&1&-\Delta&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix}.$$

G-seminorm. By inspection, n satisfies nonnegativity and identity. For symmetry, we have

$$\begin{aligned}
&n\left(\begin{bmatrix}1&-m&-K&u_1\\0&1&-\Delta&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix}^{-1}\right)\\
&= n\left(\begin{bmatrix}1&m&K+\Delta m&-u_1-u_2m\\0&1&\Delta&-u_2\\0&0&1&0\\0&0&0&1\end{bmatrix}\right)\\
&= \max\left(\frac{|-m|}{2^\kappa B}, \frac{|-\Delta|}{\ell}, \frac{|-u_2|}{\ell}, \sqrt{\frac{|-2K-\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{|-2u_1-u_2m|}{(2^\kappa B+2)\ell}}\right)\\
&= \max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right)\\
&= n\left(\begin{bmatrix}1&-m&-K&u_1\\0&1&-\Delta&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix}\right)
\end{aligned}$$

For subadditivity, we have

$$\begin{bmatrix}1&-m&-K&u_1\\0&1&-\Delta&u_2\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&-m'&-K'&u_1'\\0&1&-\Delta'&u_2'\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&-m-m'&-K-K'+\Delta' m&u_1+u_1'-u_2'm\\0&1&-\Delta-\Delta'&u_2+u_2'\\0&0&1&0\\0&0&0&1\end{bmatrix},$$

and

$$\begin{aligned}
\frac{|m+m'|}{2^\kappa B} &\le \frac{|m|}{2^\kappa B} + \frac{|m'|}{2^\kappa B}\\
\frac{|\Delta+\Delta'|}{\ell} &\le \frac{|\Delta|}{\ell} + \frac{|\Delta'|}{\ell}\\
\frac{|u_2+u_2'|}{\ell} &\le \frac{|u_2|}{\ell} + \frac{|u_2'|}{\ell}.
\end{aligned}$$

Next, we prove a corresponding inequality for $\sqrt{\frac{|2u_1+u_2m|}{2^\kappa B\ell}}$. We want to show that

$$\begin{aligned}
&\max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right) + \max\left(\frac{|m'|}{2^\kappa B}, \frac{|u_2'|}{\ell}, \sqrt{\frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell}}\right)\\
&\ge \sqrt{\frac{|2u_1+2u_1'-2u_2'm+(u_2+u_2')(m+m')|}{(2^\kappa B+2)\ell}}.
\end{aligned}$$

We square both sides, then prove the inequality.

$$
\begin{aligned}
&\left(\max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right) + \max\left(\frac{|m'|}{2^\kappa B}, \frac{|u_2'|}{\ell}, \sqrt{\frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell}}\right)\right)^2 \\
&= \max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right)^2 + \max\left(\frac{|m'|}{2^\kappa B}, \frac{|u_2'|}{\ell}, \sqrt{\frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell}}\right)^2 \\
&+ \max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right) \cdot \max\left(\frac{|m'|}{2^\kappa B}, \frac{|u_2'|}{\ell}, \sqrt{\frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell}}\right) \\
&+ \max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell}}\right) \cdot \max\left(\frac{|m'|}{2^\kappa B}, \frac{|u_2'|}{\ell}, \sqrt{\frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell}}\right) \\
&\geq \frac{|2u_1+u_2m|}{(2^\kappa B+2)\ell} + \frac{|2u_1'+u_2'm'|}{(2^\kappa B+2)\ell} + \frac{|m|}{2^\kappa B}\frac{|u_2'|}{\ell} + \frac{|u_2|}{\ell}\frac{|m'|}{2^\kappa B} \\
&\geq \frac{|2u_1+u_2m+2u_1'+u_2'm'-u_2'm+u_2m'|}{(2^\kappa B+2)\ell} \\
&= \frac{|2u_1+2u_1'-2u_2'm+(u_2+u_2')(m+m')|}{(2^\kappa B+2)\ell}
\end{aligned}
$$

Finally, we need to prove a similar bound on $\sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}$:

$$
\begin{aligned}
&\max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&+ \max\left(\frac{|m'|}{2^\kappa B}, \frac{|\Delta'|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&\geq \sqrt{\frac{|2K+2K'-2\Delta' m+(\Delta+\Delta')(m+m')|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}.
\end{aligned}
$$

If we let $f(z) = z^2 + 2\sqrt{\frac{q}{B\ell}} \cdot z = z(z + 2\sqrt{\frac{q}{B\ell}})$, then we have

$$
f\left(\sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) = \frac{|2K+\Delta m|}{2^\kappa B\ell}.
$$

Therefore, we apply f (an increasing function) to both sides to simplify the inequality. We get

$$
\begin{aligned}
&f\left(\begin{array}{l}\max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\ + \max\left(\frac{|m'|}{2^\kappa B}, \frac{|\Delta'|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\end{array}\right) \\
&= f\left(\max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\right) \\
&+ f\left(\max\left(\frac{|m'|}{2^\kappa B}, \frac{|\Delta'|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\right) \\
&+ 2 \cdot \max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)
\end{aligned}
$$

$$
\begin{aligned}
&\cdot \max\left(\frac{|m'|}{2^\kappa B}, \frac{|\Delta'|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&\geq \frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{|2K'+\Delta' m'|}{2^\kappa B\ell} + \frac{|m|}{2^\kappa B}\frac{|\Delta'|}{\ell} + \frac{|\Delta|}{\ell}\frac{|m'|}{2^\kappa B} \\
&\geq \frac{|2K+\Delta m+2K'+\Delta' m'-\Delta' m+\Delta m'|}{2^\kappa B\ell} \\
&= \frac{|2K+2K'-2\Delta' m+(\Delta+\Delta')(m+m')|}{2^\kappa B\ell}
\end{aligned}
$$

This completes the proof of subadditivity.

G-seminorm bounds on components. Next, we need to show that decomposition respects the norm. Let $\boldsymbol{R}_{\mathsf{in}} = \overline{\boldsymbol{R}}_{\mathsf{in}} \star \overline{\boldsymbol{R}}_{\mathsf{MAC}} \star \boldsymbol{R}$ (for $\overline{\boldsymbol{R}}_{\mathsf{in}} \in \overline{\mathsf{R}}_{\mathsf{in}}$, $\overline{\boldsymbol{R}}_{\mathsf{MAC}} \in \overline{\mathsf{R}}_{\mathsf{MAC}}$, and $\boldsymbol{R} \in \mathsf{R}_{\mathsf{pk}}$). We have

$$
\frac{|\Delta|}{\ell} \leq \max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K+\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \leq n(m, K, \Delta, u_1, u_2)
$$

so

$$
\begin{aligned}
n(\boldsymbol{R}) \leq n(\overline{\boldsymbol{R}}_{\mathsf{MAC}} \star \boldsymbol{R}) \leq n(\boldsymbol{R}_{\mathsf{in}}) \\
n(\mathsf{key}(\boldsymbol{R}_{\mathsf{in}})) \leq n(\mathsf{in}(\boldsymbol{R}_{\mathsf{in}})) \leq n(\boldsymbol{R}_{\mathsf{in}}).
\end{aligned}
$$

Since $n((\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2})) = \max(n(\boldsymbol{R}_{\mathsf{in}\text{-}1}), n(\boldsymbol{R}_{\mathsf{in}\text{-}2}))$, these inequalities imply

$$
n(\mathsf{key}(\rho_c)) \leq n(\rho_c) \qquad n(\mathsf{in}_\sigma(\rho_c)) \leq n(\rho_c) \qquad n(\overline{\rho}_1 \star \overline{\rho}_2 \star \Delta) \leq n(\rho_c).
$$

We also bound the individual coefficients (m, K, Δ, u_1, u_2) for any $\boldsymbol{R}_{\mathsf{in}}$ and $\beta \geq n(\boldsymbol{R}_{\mathsf{in}})$, as these bounds will be useful later on. Trivially, we have

$$
|m| \leq 2^\kappa B\beta \tag{3}
$$

$$
|\Delta| \leq \ell\beta \tag{4}
$$

$$
|u_2| \leq \ell\beta. \tag{5}
$$

Next, for u_1 we have

$$
\begin{aligned}
\beta^2 &\geq \max\left(\frac{|m|}{2^\kappa B}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2u_1+u_2 m|}{(2^\kappa B+2)\ell}}\right)^2 \\
&= \max\left(\frac{m^2}{2^{2\kappa}B^2}, \frac{u_2^2}{\ell^2}, \frac{|2u_1+u_2 m|}{(2^\kappa B+2)\ell}\right) \\
&\geq \max\left(\frac{|mu_2|}{2^\kappa B\ell}, \frac{|2u_1+u_2 m|}{(2^\kappa B+2)\ell}\right) \\
&\geq \max\left(\frac{|mu_2|}{(2^\kappa B+2)\ell}, \frac{|2u_1+u_2 m|}{(2^\kappa B+2)\ell}\right) \\
&\geq \frac{|mu_2|+|2u_1+u_2 m|}{2(2^\kappa B+2)\ell} \\
&\geq \frac{|-mu_2+2u_1+u_2 m|}{2(2^\kappa B+2)\ell} \\
&= \frac{|u_1|}{(2^\kappa B+2)\ell}.
\end{aligned}
$$

Therefore,

$$|u_1| \leq (2^\kappa B + 2)\ell\beta^2. \tag{6}$$

Finally, we need to bound K. Let $f(z) = z^2 + 2\sqrt{\frac{q}{B\ell}} \cdot z$. Since f is increasing,

$$\begin{aligned}
f(\beta) &\geq f\left(\max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \sqrt{\frac{|2K + \Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\right) \\
&= \max\left(f\left(\frac{m}{2^\kappa B}\right), f\left(\frac{\Delta}{\ell}\right), f\left(\sqrt{\frac{|2K + \Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\right) \\
&\geq \max\left(\frac{m^2}{2^{2\kappa}B^2}, \frac{\Delta^2}{\ell^2}, \frac{|2K + \Delta m|}{2^\kappa B\ell}\right) \\
&\geq \max\left(\frac{|m\Delta|}{2^\kappa B\ell}, \frac{|2K + \Delta m|}{2^\kappa B\ell}\right) \\
&\geq \frac{|m\Delta| + |2K + \Delta m|}{2^{\kappa+1} B\ell} \\
&\geq \frac{|-m\Delta + 2K + \Delta m|}{2^{\kappa+1} B\ell} \\
&= \frac{|K|}{2^\kappa B\ell}.
\end{aligned}$$

Therefore,

$$|K| \leq 2^\kappa B\ell\left(\beta^2 + 2\sqrt{\frac{q}{B\ell}} \cdot \beta\right) = 2^\kappa B\ell\beta^2 + 2^{\kappa+1}\sqrt{qB\ell} \cdot \beta. \tag{7}$$

Group actions Rerand and $\mathsf{Re}_{\mathsf{MAC}}$. Rerand is a group action, as required by Definition 8, because of the associativity of matrix multiplication. $\mathsf{Re}_{\mathsf{MAC}}$ is also a group action, because of associativity and because taking the inverse transpose is a homomorphism.

Structure of Rerand and $\mathsf{Re}_{\mathsf{MAC}}$. We must show that Rerand and $\mathsf{Re}_{\mathsf{MAC}}$ follow the structure assigned to them by Definition 8. For Rerand, this is equivalent to showing that the Rerand is compatible with $\mathsf{Re}_{\mathsf{pk}}$ and $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$, and that $\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\boldsymbol{R}, [\overline{\mathsf{pk}}_\sigma\ \mathsf{pk}\ -\mathsf{g}]) = [\overline{\mathsf{pk}}'_\sigma\ \mathsf{pk}'\ -\mathsf{g}']^\mathsf{T}$ satisfies $\overline{\mathsf{pk}}'_\sigma = \overline{\mathsf{pk}}_\sigma$ when $\boldsymbol{R} \in \mathsf{R}_{\mathsf{pk}}$.

Rerand is compatible with $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$, i.e., the wire public keys inside the rerandomized ciphertext $\mathbf{C}' = \mathsf{Rerand}(\rho_c, \mathbf{C})$ are $\mathsf{pk}'_\sigma = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\mathsf{in}_\sigma(\rho_c), \mathsf{pk}_\sigma)$:

$$\begin{aligned}
&\mathsf{Rerand}(\rho_c = (\boldsymbol{R}_{\mathsf{in\text{-}1}}, \boldsymbol{R}_{\mathsf{in\text{-}2}}), c = \mathbf{C}) \\
&= \underbrace{\begin{bmatrix} 1 & -m_1 & -K_1 & u_{1,1} \\ 0 & 1 & -\rho_1 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R}_{\mathsf{in\text{-}1}}} \cdot \begin{bmatrix} \overline{\boldsymbol{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & -m_2 & -K_2 & u_{2,1} \\ 0 & 1 & -\rho_2 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T}}_{\boldsymbol{R}_{\mathsf{in\text{-}2}}} \\
&= \begin{bmatrix} \begin{bmatrix} 1 & -m_1 & -K_1 & u_{1,1} \\ 0 & 1 & -\rho_1 & u_{1,2} \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \overline{\boldsymbol{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -m_2 & -K_2 & u_{2,1} \\ 0 & 1 & -\rho_2 & u_{2,2} \\ 0 & 0 & 1 & 0 \end{bmatrix}^\mathsf{T} & \begin{bmatrix} 1 & -m_1 & -K_1 \\ 0 & 1 & -\rho_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} \cdot \begin{bmatrix} 1 & -m_2 & -K_2 \\ 0 & 1 & -\rho_2 \\ 0 & 0 & 1 \end{bmatrix}^\mathsf{T} & 0 \end{bmatrix}
\end{aligned}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & -m_1 & -K_1 \\ 0 & 1 & -\rho_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \overline{\boldsymbol{C}} \cdot \begin{bmatrix} 1 & -m_2 & -K_2 \\ 0 & 1 & -\rho_2 \\ 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} + \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ 0 \end{bmatrix} \mathsf{pk}_2'^{\mathsf{T}} + \mathsf{pk}_1' \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ 0 \end{bmatrix}^{\mathsf{T}} & \mathsf{pk}_1' \\ \mathsf{pk}_2'^{\mathsf{T}} & 0 \end{bmatrix}.$$

$\mathsf{Re}_{\mathsf{pk\text{-}wire}}$ is compatible with $\mathsf{Re}_{\mathsf{pk}}$, i.e., the global public key inside the rerandomized wire public key $\mathsf{pk}'_\sigma = \mathsf{Rerand}(\boldsymbol{R}_\sigma, \mathsf{pk}_\sigma)$ is $\mathsf{pk}' = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\mathsf{key}(\boldsymbol{R}_\sigma), \mathsf{pk})$:

$$\begin{bmatrix} \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\boldsymbol{R}_\sigma, \mathsf{pk}_\sigma) \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R}_\sigma} \cdot \begin{bmatrix} \overline{\mathsf{pk}}_\sigma \\ \mathsf{pk} \\ -\mathsf{g} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \overline{\mathsf{pk}}_\sigma - m \cdot \mathsf{pk} + K \cdot \mathsf{g} \\ \mathsf{pk} + \Delta \cdot \mathsf{g} \\ -\mathsf{g} \\ 0 \end{bmatrix} = \begin{bmatrix} \overline{\mathsf{pk}}_\sigma - m \cdot \mathsf{pk} + K \cdot \mathsf{g} \\ \mathsf{pk}' \\ -\mathsf{g} \\ 0 \end{bmatrix}$$

We must also check that $\overline{\mathsf{Re}_{\mathsf{pk\text{-}wire}}}$ is doesn't depend on the global key, i.e., that $\overline{\mathsf{pk}}'_\sigma = \overline{\mathsf{pk}}_\sigma$ when $\boldsymbol{R}_\sigma \in \mathsf{R}_{\mathsf{pk}}$. We have $m = K = 0$ when $\boldsymbol{R}_\sigma \in \mathsf{R}_{\mathsf{pk}}$, so this holds by inspection of the equation above.

Finally, we must check that $\mathsf{Re}_{\mathsf{MAC}}$ defined in Algorithm 3 satisfies the structure prescribed by Definition 8:

$\mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma))$:
$\overline{\boldsymbol{x}}'_\sigma = \overline{\boldsymbol{x}}_\sigma + \mathsf{offset}(\rho_\sigma)$
$\boldsymbol{X}'_\sigma = \boldsymbol{X}_\sigma + \mathsf{MAC}(\rho_\sigma, \overline{\boldsymbol{x}}_\sigma)$
return $(\overline{\boldsymbol{x}}'_\sigma, \boldsymbol{X}'_\sigma)$

Let

$$\mathsf{MAC}\left(\begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \overline{x}\right) = K + \Delta(\overline{x} + m)$$

Then,

$$\begin{bmatrix} 1 \\ \mathsf{Re}_{\mathsf{MAC}}(\boldsymbol{R}_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)) \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-\mathsf{T}} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma \\ \boldsymbol{X}_\sigma \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ m & 1 & 0 & 0 \\ K + m\Delta & \Delta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma \\ \boldsymbol{X}_\sigma \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma + m \\ \boldsymbol{X}_\sigma + K + \Delta(\overline{\boldsymbol{x}}_\sigma + m) \\ 0 \end{bmatrix} \qquad = \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma + \mathsf{offset}(\boldsymbol{R}_\sigma) \\ \boldsymbol{X}_\sigma + \mathsf{MAC}(\boldsymbol{R}_\sigma, \overline{\boldsymbol{x}}_\sigma) \\ 0 \end{bmatrix}.$$

Valid wires subgroup. We have

$$\mathcal{W}_{\mathsf{pk}} = \left\{ (\overline{\mathsf{pk}}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \;\middle|\; [1\ \overline{\boldsymbol{x}}_\sigma\ \boldsymbol{X}_\sigma] \cdot \begin{bmatrix} \overline{\mathsf{pk}}_\sigma \\ \mathsf{pk} \\ -\mathsf{g} \end{bmatrix} = 0 \right\}.$$

We need to show that this forms a group. The equation can be written as

$$\overline{\mathsf{pk}}_\sigma + \overline{\boldsymbol{x}}_\sigma \cdot \mathsf{pk} - \boldsymbol{X}_\sigma \cdot \mathsf{g} = 0.$$

where pk and g are constants. This is linear in $(\overline{\mathsf{pk}}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma)$ (i.e., it is a homomorphism), so therefore the solutions (i.e., the kernel) form a group under addition.

Next, we need to show that $\mathcal{W}$ is preserved by the group action. For all $(\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \in \mathcal{W}$, we need to check that

$$(\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\rho_\sigma, \mathsf{pk}_\sigma), \mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma))) \in \mathcal{W}.$$

Plugging in the definitions, this is equivalent to

$$\begin{aligned}
\left(\boldsymbol{R}_\sigma^{-\mathsf{T}} \cdot \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma \\ \boldsymbol{X}_\sigma \\ 0 \end{bmatrix}\right)^{\mathsf{T}} \boldsymbol{R}_\sigma \cdot \begin{bmatrix} \mathsf{pk}_\sigma \\ 0 \end{bmatrix} &= 0 \\
\begin{bmatrix} 1 & \overline{\boldsymbol{x}}_\sigma & \boldsymbol{X}_\sigma & 0 \end{bmatrix} \cdot \boldsymbol{R}_\sigma^{-1} \boldsymbol{R}_\sigma \cdot \begin{bmatrix} \mathsf{pk}_\sigma \\ 0 \end{bmatrix} &= 0 \\
\begin{bmatrix} 1 & \overline{\boldsymbol{x}}_\sigma & \boldsymbol{X}_\sigma \end{bmatrix} \cdot \mathsf{pk}_\sigma &= 0,
\end{aligned}$$

which is exactly $(\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \in \mathcal{W}$.

Linearity of Eval. Eval satisfies linearity because it applies DLog to a linear function of the ciphertexts $\overline{C}$, and DLog is a homomorphism from $\mathbb{F}$ to $\mathbb{Z}_q$.

Linearity of Rerand. We divide the linearity proof for Rerand into two steps. First, if $\rho_c \in \langle \mathsf{R}_{\mathsf{MAC},1} \cup \mathsf{R}_{\mathsf{MAC},2} \rangle$ then $\rho_c = (\boldsymbol{R}_{\mathsf{in}\text{-}1}, \boldsymbol{R}_{\mathsf{in}\text{-}2})$, where the last row and column of $\boldsymbol{R}_{\mathsf{in}\text{-}\sigma}$ (for $\sigma \in [2]$) are the same in an identity matrix. Therefore,

$$\begin{aligned}
\mathsf{Rerand}(\rho_c, \mathbf{C} + \mathbf{C}') &= \boldsymbol{R}_{\mathsf{in}\text{-}1} \begin{bmatrix} \overline{\mathbf{C}} + \overline{\mathbf{C}'} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} \boldsymbol{R}_{\mathsf{in}\text{-}2}{}^{\mathsf{T}} \\
&= \boldsymbol{R}_{\mathsf{in}\text{-}1} \begin{bmatrix} \overline{\mathbf{C}} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & 0 \end{bmatrix} \boldsymbol{R}_{\mathsf{in}\text{-}2}{}^{\mathsf{T}} + \boldsymbol{R}_{\mathsf{in}\text{-}1} \begin{bmatrix} \overline{\mathbf{C}'} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & 0 \end{bmatrix} \boldsymbol{R}_{\mathsf{in}\text{-}2}{}^{\mathsf{T}} + \boldsymbol{R}_{\mathsf{in}\text{-}1} \begin{bmatrix} \mathbf{0} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} \boldsymbol{R}_{\mathsf{in}\text{-}2}{}^{\mathsf{T}} \\
&= \mathsf{Rerand}(\rho_c, \mathbf{C}) + \mathsf{Rerand}(\rho_c, \mathbf{C}')
\end{aligned}$$

Second, if $\overline{\rho}_c \in \overline{\mathsf{R}}$, then $\overline{\rho}_c = (\overline{\boldsymbol{R}}_{\mathsf{in}\text{-}1}, \overline{\boldsymbol{R}}_{\mathsf{in}\text{-}2})$, and we have

$$\begin{aligned}
\mathsf{Rerand}(\overline{\rho}_c, \mathbf{C}) &= \overline{\boldsymbol{R}}_{\mathsf{in}\text{-}1} \begin{bmatrix} \overline{\mathbf{C}} + \overline{\mathbf{C}'} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} \overline{\boldsymbol{R}}_{\mathsf{in}\text{-}2}{}^{\mathsf{T}} \\
&= \begin{bmatrix} 1 & 0 & 0 & u_{1,1} \\ 0 & 1 & 0 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \overline{\mathbf{C}} + \overline{\mathbf{C}'} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & u_{2,1} \\ 0 & 1 & 0 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} \\
&= \begin{bmatrix} \overline{\mathbf{C}} + \overline{\mathbf{C}'} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} + \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \mathsf{pk}_2 \\ 0 \end{bmatrix}^{\mathsf{T}} + \begin{bmatrix} \mathsf{pk}_1 \\ 0 \end{bmatrix} \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ 0 \\ 0 \end{bmatrix}^{\mathsf{T}},
\end{aligned}$$

so it is just adding an offset to $\mathbf{C}$, and this offset is a linear function of $\overline{\rho}_c$. Therefore,

$$\mathsf{Rerand}(\overline{\rho}_c \star \overline{\rho}'_c, \mathbf{C} + \mathbf{C}') = \mathsf{Rerand}(\overline{\rho}_c, \mathbf{C}) + \mathsf{Rerand}(\overline{\rho}'_c, \mathbf{C}').$$

Linearity of $\mathsf{Re}_{\mathsf{pk\text{-}wire}}$.

$$\mathsf{Re}_{\mathsf{pk\text{-}wire}}\left(\left[\begin{bmatrix}1 & -m & -K & 0\\ 0 & 1 & -\Delta & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\right], \mathsf{pk}_\sigma + \mathsf{pk}'_\sigma\right) = \begin{bmatrix}1 & -m & -K\\ 0 & 1 & -\Delta\\ 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\overline{\mathsf{pk}}_\sigma\\ \mathsf{pk}\\ -\mathsf{g}\end{bmatrix}$$
$$= \begin{bmatrix}1 & -m & -K\\ 0 & 1 & -\Delta\\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}\overline{\mathsf{pk}}_\sigma - m\cdot\mathsf{pk} + K\cdot\mathsf{g}\\ \mathsf{pk} + \Delta\cdot\mathsf{g}\\ -\mathsf{g}\end{bmatrix},$$

so $\overline{\mathsf{pk}}_\sigma$ is unaffected by $\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\boldsymbol{R},\cdot)$ for $\boldsymbol{R}\in\mathsf{R}_{\mathsf{pk}}$. $\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\boldsymbol{R}}_\sigma,\cdot)$ for $\overline{\boldsymbol{R}}_\sigma\in\overline{\mathsf{R}}_{\mathsf{MAC}}$ just adds an offset to $\overline{\mathsf{pk}}_\sigma$, and this offset is linear in $\overline{\boldsymbol{R}}_\sigma$. Therefore,

$$\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\boldsymbol{R}}_\sigma \star \overline{\boldsymbol{R}}'_\sigma \star \boldsymbol{R}, \mathsf{pk}_\sigma + \mathsf{pk}'_\sigma) = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\boldsymbol{R}}_\sigma \star \boldsymbol{R}, \mathsf{pk}_\sigma) + \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\boldsymbol{R}}'_\sigma \star \boldsymbol{R}, \mathsf{pk}'_\sigma).$$

Linearity of MAC (and $\mathsf{Re}_{\mathsf{MAC}}$). We split the argument into to steps. For $\boldsymbol{R}\in\mathsf{R}_{\mathsf{pk}}$, represented by $\Delta\in\mathbb{Z}$, we have

$$\mathsf{MAC}(\boldsymbol{R}, \boldsymbol{x}_\sigma + \boldsymbol{x}'_\sigma) = \Delta\cdot(\boldsymbol{x}_\sigma + \boldsymbol{x}'_\sigma) = \mathsf{MAC}(\boldsymbol{R},\boldsymbol{x}_\sigma) + \mathsf{MAC}(\boldsymbol{R},\boldsymbol{x}'_\sigma)$$

For $\overline{\boldsymbol{R}}_\sigma, \overline{\boldsymbol{R}}'_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC}}$, represented by $m_\sigma, K_\sigma, m'_\sigma, K'_\sigma\in\mathbb{Z}$, we have

$$\mathsf{MAC}(\overline{\boldsymbol{R}}_\sigma \star \overline{\boldsymbol{R}}'_\sigma \star \boldsymbol{R}, \boldsymbol{x}_\sigma + \boldsymbol{x}'_\sigma) = K_\sigma + K'_\sigma = \mathsf{MAC}(\overline{\boldsymbol{R}}_\sigma, \boldsymbol{x}_\sigma) + \mathsf{MAC}(\overline{\boldsymbol{R}}'_\sigma, \boldsymbol{x}'_\sigma).$$

Bound on $n(\mathcal{G}(1^\kappa))$. From Definition 8, the $\rho_c = (\boldsymbol{R}_{\mathsf{in\text{-}1}}, \boldsymbol{R}_{\mathsf{in\text{-}2}})$ output by $\mathcal{G}$ follow the distribution:

$$\boldsymbol{R}_{\mathsf{in}\text{-}\sigma} \leftarrow \left\{ \begin{bmatrix}1 & -m & -K+\Delta m & u_1\\ 0 & 1 & -\Delta & u_2\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \middle| \begin{array}{r} \Delta, u_1, u_2 \leftarrow [\ell]\\ m \leftarrow [2^\kappa B]\\ K \leftarrow \left[2^\kappa\sqrt{qB\ell}\right]\end{array} \right\} \tag{8}$$

Therefore,

$$\begin{aligned}
n(\boldsymbol{R}_{\mathsf{in}\text{-}\sigma}) &= \max\left(\frac{|m|}{2^\kappa B}, \frac{|\Delta|}{\ell}, \frac{|u_2|}{\ell}, \sqrt{\frac{|2K-\Delta m|}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{|2u_1+u_2 m|}{(2^\kappa B+2)\ell}}\right)\\
&\le \max\left(1,1,1, \sqrt{\frac{\max(2^{\kappa+1}\sqrt{qB\ell}, 2^\kappa B\ell)}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{|2\ell+2^\kappa B\ell|}{(2^\kappa B+2)\ell}}\right)\\
&= \max\left(1, \sqrt{\frac{2^{\kappa+1}\sqrt{qB\ell}}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{\frac{2^\kappa B\ell}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, 1\right)\\
&= \max\left(1, \sqrt{2\sqrt{\frac{q}{B\ell}} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}, \sqrt{1+\frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right)\\
&= \max\left(1, \frac{2\sqrt{\frac{q}{B\ell}}}{\sqrt{2\sqrt{\frac{q}{B\ell}} + \frac{q}{B\ell}} + \sqrt{\frac{q}{B\ell}}}, \frac{1}{\sqrt{1+\frac{q}{B\ell}} + \sqrt{\frac{q}{B\ell}}}\right)\\
&= 1
\end{aligned}$$

Similarly, we need to bound the G-seminorm of the samples from $\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta), \mathcal{G}^*_{\mathsf{MAC}}(1^\kappa,\beta), \mathcal{G}^*_{\mathsf{Enc}}$.

From their definitions, we have

$$
\begin{aligned}
n(\mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)) &\le \frac{2^\kappa\beta\ell}{\ell} = 2^\kappa\beta \\
n(\mathcal{G}^*_{\mathsf{MAC}}(1^\kappa,\beta)) &\le \max\left(\frac{2^{2\kappa}\beta B}{2^\kappa B}, \sqrt{\frac{2\cdot(2^{2\kappa}\beta^2 B\ell + 2^{2\kappa+1}\beta\sqrt{qB\ell})}{2^\kappa B\ell} + \frac{q}{B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&= \max\left(2^\kappa\beta, \sqrt{\frac{2^{2\kappa+1}\beta^2 B\ell + 2^{2\kappa+2}\beta\sqrt{qB\ell} + 2^\kappa q}{2^\kappa B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&\le \max\left(2^\kappa\beta, \sqrt{\frac{2^\kappa(2^{\kappa+1}\beta\sqrt{B\ell} + \sqrt{q})^2}{2^\kappa B\ell}} - \sqrt{\frac{q}{B\ell}}\right) \\
&= 2^{\kappa+1}\beta \\
n(\mathcal{G}^*_{\mathsf{Enc}}) &\le \max\left(\frac{2^\kappa\beta\ell}{\ell}, \sqrt{\frac{2\cdot 2^{2\kappa}\beta^2 B\ell}{(2^\kappa B+2)\ell}}\right) \\
&\le \max\left(2^\kappa\beta, \sqrt{2\cdot 2^\kappa\beta^2}\right) \\
&= 2^\kappa\beta.
\end{aligned}
$$

Therefore, we set $\beta_{\mathrm{mask}} = 2^{\kappa+1}$.

Correctness. We state correctness equivalently as follows. For every $f \in \mathcal{F}_{\mathsf{pp}}$, input $\boldsymbol{x} \in \mathcal{X}^2$, and randomness $\rho_c \in \mathsf{R}$, if

$$
\begin{aligned}
\mathbf{C} &= \mathsf{Compile}(f) \\
\{(\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) &= \mathsf{Re}_{\mathsf{MAC}}(\mathsf{in}_\sigma(\rho_c), \boldsymbol{x}_\sigma, 0)\}_{\sigma\in[2]} \\
\mathbf{C}' &= \mathsf{Rerand}(\rho_c, \mathbf{C})
\end{aligned}
$$

then

$$
f(\boldsymbol{x}) = \mathsf{Eval}(\mathbf{C}', \boldsymbol{X}, \overline{\boldsymbol{x}}).
$$

Let $\boldsymbol{F}$ be the matrix representing the bilinear function f, i.e., $f(\boldsymbol{x}) = [1\ \boldsymbol{x}_1]\boldsymbol{F}[1\ \boldsymbol{x}_2]^\mathsf{T}$, let $\rho_c = (\boldsymbol{R}_{\text{in-1}}, \boldsymbol{R}_{\text{in-2}})$, and for $\sigma \in [2]$ write $\boldsymbol{R}_{\text{in-}\sigma} = \overline{\boldsymbol{R}}_{\text{in-}\sigma} \star \boldsymbol{R}_\sigma$ where $\overline{\boldsymbol{R}}_{\text{in-}\sigma} \in \overline{\mathsf{R}}_{\text{in-}\sigma}$ and $\boldsymbol{R}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$. Let $\overline{\boldsymbol{R}}_{\text{in-}\sigma}{}' = \boldsymbol{R}_\sigma^{-1} \star \overline{\boldsymbol{R}}_{\text{in-}\sigma} \star \boldsymbol{R}_\sigma$, which is still in $\overline{\mathsf{R}}_{\text{in-}\sigma}$ because its a normal subgroup. We now substitute the definitions for Rerand, $\mathsf{Re}_{\mathsf{MAC}}$, Eval, and $\mathsf{Compile}$:

$$
\begin{aligned}
\mathbf{C}' &= \boldsymbol{R}_{\text{in-1}} \cdot \mathbf{C} \cdot \boldsymbol{R}_{\text{in-2}}{}^\mathsf{T} \\
\begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_\sigma \\ \boldsymbol{X}_\sigma \\ 0 \end{bmatrix} &= \boldsymbol{R}_\sigma^{-\mathsf{T}} \begin{bmatrix} 1 \\ \boldsymbol{x}_\sigma \\ 0 \\ 0 \end{bmatrix} \\
\mathsf{Eval}(\mathbf{C}', \boldsymbol{X}, \overline{\boldsymbol{x}}) &= \mathsf{DLog}\left(\begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_1 \\ \boldsymbol{X}_1 \\ 0 \end{bmatrix}^\mathsf{T} \cdot \mathbf{C}' \cdot \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}_2 \\ \boldsymbol{X}_2 \\ 0 \end{bmatrix}\right) \\
&= \mathsf{DLog}\left(\begin{bmatrix} 1 \\ \boldsymbol{x}_1 \\ 0 \\ 0 \end{bmatrix}^\mathsf{T} \cdot \boldsymbol{R}_1^{-1} \cdot \boldsymbol{R}_{\text{in-1}} \cdot \mathbf{C} \cdot \boldsymbol{R}_{\text{in-2}}{}^\mathsf{T} \cdot \boldsymbol{R}_2^{-\mathsf{T}} \cdot \begin{bmatrix} 1 \\ \boldsymbol{x}_2 \\ 0 \\ 0 \end{bmatrix}\right)
\end{aligned}
$$

$$= \mathsf{DLog}\left(\begin{bmatrix}1\\ \boldsymbol{x}_1\\ 0\\ 0\end{bmatrix}^{\mathsf{T}} \cdot \overline{\boldsymbol{R}}_{\text{in-1}}{}' \cdot \mathbf{C} \cdot \overline{\boldsymbol{R}}_{\text{in-2}}{}'^{\mathsf{T}} \cdot \begin{bmatrix}1\\ \boldsymbol{x}_2\\ 0\\ 0\end{bmatrix}\right)$$

$$\mathbf{C} = \begin{bmatrix}\boldsymbol{F}\cdot \mathsf{f} & \mathbf{0} & \mathbf{0}\\ \mathbf{0}^{\mathsf{T}} & 0 & -\mathsf{g}\\ \mathbf{0}^{\mathsf{T}} & -\mathsf{g} & 0\end{bmatrix}$$

$$\mathsf{Eval}(\mathbf{C}', \boldsymbol{X}, \overline{\boldsymbol{x}}) = \mathsf{DLog}\left(\begin{bmatrix}1\\ \boldsymbol{x}_1\\ 0\\ 0\end{bmatrix}^{\mathsf{T}} \cdot \begin{bmatrix}1 & 0 & 0 & u'_{1,1}\\ 0 & 1 & 0 & u'_{1,2}\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{F}\cdot \mathsf{f} & \mathbf{0} & \mathbf{0}\\ \mathbf{0}^{\mathsf{T}} & 0 & -\mathsf{g}\\ \mathbf{0}^{\mathsf{T}} & -\mathsf{g} & 0\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 0 & u'_{2,1}\\ 0 & 1 & 0 & u'_{2,2}\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}^{\mathsf{T}} \cdot \begin{bmatrix}1\\ \boldsymbol{x}_2\\ 0\\ 0\end{bmatrix}\right)$$

$$= \mathsf{DLog}\left(\begin{bmatrix}1\\ \boldsymbol{x}_1\\ 0\\ u'_{1,1} + u'_{1,2}\boldsymbol{x}_1\end{bmatrix}^{\mathsf{T}} \cdot \begin{bmatrix}\boldsymbol{F}\cdot \mathsf{f} & \mathbf{0} & \mathbf{0}\\ \mathbf{0}^{\mathsf{T}} & 0 & -\mathsf{g}\\ \mathbf{0}^{\mathsf{T}} & -\mathsf{g} & 0\end{bmatrix} \cdot \begin{bmatrix}1\\ \boldsymbol{x}_2\\ 0\\ u'_{2,1} + u'_{2,2}\boldsymbol{x}_2\end{bmatrix}\right)$$

$$= \mathsf{DLog}\left(\begin{bmatrix}1\\ \boldsymbol{x}_1\end{bmatrix}^{\mathsf{T}} \cdot \boldsymbol{F} \cdot \mathsf{f} \cdot \begin{bmatrix}1\\ \boldsymbol{x}_2\end{bmatrix} - (u'_{1,1} + u'_{1,2}\boldsymbol{x}_1)\cdot \mathsf{g} \cdot 0 - 0 \cdot \mathsf{g} \cdot (u'_{2,1} + u'_{2,2}\boldsymbol{x}_2)\right)$$

$$= f(\boldsymbol{x}).$$

Affine boundedness. Note that $D = \{x \in \mathcal{X} \mid \iota(x) = x\} = [-q/2, q/2) \cap \mathbb{Z}$, so if $|x| < \frac{q}{2}$ then $x \in D$. Therefore, we need to prove size bounds on $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)$ and $X_\sigma = \mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)$. In particular, when $n(\rho_\sigma) \le T$ we will show:

1. $|\boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)| \le B(2^\kappa T + 1)$.
2. $|\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)| \le 2^{\kappa+1}(B\ell T^2 + (\sqrt{qB\ell} + B\ell)\cdot T)$.

Since the affine requirement follows by inspection, proving these two bounds is enough to prove affine boundedness.

First, we have $\boldsymbol{x}_\sigma \in \mathcal{X}' = [-B, B]$, and by Equation (3), $\mathsf{offset}(\rho_\sigma) \le 2^\kappa BT$. Therefore,

$$|\boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)| \le B(2^\kappa T + 1) \tag{9}$$

Second, by Equations (4) and (7),

$$|\Delta| \le \ell T$$
$$|K_\sigma| \le 2^\kappa B\ell T^2 + 2^{\kappa+1}\sqrt{qB\ell}\cdot T,$$

so using Equation (9),

$$\begin{aligned}|\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)| &\le 2^\kappa B\ell T^2 + 2^{\kappa+1}\sqrt{qB\ell}\cdot T + \ell T B(2^\kappa T + 1)\\ &= 2^{\kappa+1}B\ell T^2 + 2^{\kappa+1}\sqrt{qB\ell}\cdot T + B\ell T.\end{aligned}$$

Now, let us plug in $T = 2^{-\kappa-3}\sqrt{\frac{q}{B\ell}}$:

$$\begin{aligned}|\boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)| &\le B2^\kappa T + B\\ &\le 2^{-3}\sqrt{\frac{qB}{\ell}} + B\\ &< q/2.\\ |\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)| &\le 2^{\kappa+1}B\ell T^2 + 2^{\kappa+1}\sqrt{qB\ell}\cdot T + B\ell T\\ &\le 2^{-\kappa-5}q + 2^{-2}\cdot q + 2^{-\kappa-3}\sqrt{qB\ell}\\ &< q/2.\end{aligned}$$

Equation (4) also suffices to show that setting $\beta_{\text{mask}} = \ell$ is valid.

Masking. We must prove that

$$\begin{aligned}
\{\Delta' \star \Delta\}_{\Delta' \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{s}{\approx} \{\Delta'\}_{\Delta' \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))} \\
\{\overline{\rho'}_\sigma \star \overline{\rho}_\sigma\}_{\overline{\rho'}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{s}{\approx} \{\overline{\rho'}_\sigma\}_{\overline{\rho'}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))} \\
\{\overline{\rho'}_c \star \overline{\rho}_c\}_{\overline{\rho'}_c \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} &\overset{s}{\approx} \{\overline{\rho'}_c\}_{\overline{\rho'}_c \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))}.
\end{aligned}$$

Each of these groups is isomorphic to addition of integer vectors of some dimension, so we just need to check that the masks are sampled from wide enough ranges compared to what they are masking.

- By Equation (4), $|\Delta| \le \beta\ell$, so sampling $\Delta' \leftarrow [2^\kappa \beta\ell]$ gives statistical distance at most $2^{-\kappa}$.
- By Equations (3) and (7), $|m| \le 2^\kappa B\beta$ and $|K| \le 2^\kappa \beta^2 B\ell + 2^{\kappa+1}\beta\sqrt{qB\ell}$, so sampling $m' \leftarrow [2^{2\kappa}\beta B]$ and $K \leftarrow \left[2^{2\kappa}\beta^2 B\ell + 2^{2\kappa+1}\beta\sqrt{qB\ell}\right]$ gives statistical distance at most $2^{1-\kappa}$.
- By Equations (5) and (6), $|u_1| \le (2^\kappa B + 2)\ell\beta^2$ and $|u_2| \le \ell\beta$, so $u_1' \leftarrow [2^{2\kappa}\beta^2 B\ell]$ and $u_2' \leftarrow [2^\kappa\beta\ell]$ gives statistical distance at most $\frac{2^\kappa B+2}{2^{2\kappa}B} + 2^{-\kappa} = O(2^{-\kappa})$ per ciphertext component. Since there are two, this statistical distance is at most doubled.

Input Masking. We have $x \in \mathcal{X}' = [-B, B]$, and $m \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa) = [2^\kappa B]$ The distributions of $x + m$ and m perfectly match on all but $|x|$ of the $2^\kappa B$ possible samples, so

$$\{m + x\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa)} \overset{s}{\approx} \{m\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa)}.$$

with statistical distance at most $\frac{B}{2^\kappa B} = 2^{-\kappa}$.

Ciphertext Simulation. We start by defining the simulator $\mathcal{S}_{\mathsf{Enc}}$. $\mathcal{S}_{\mathsf{Enc}}$ is given the inputs and outputs of the function $f(\Delta)$ at the single value $\boldsymbol{x}$, and it needs to build an encryption that agrees with $f(\Delta)$ on every input $\boldsymbol{x}'$, without knowing Δ. The trick is that some entries in the ciphertext matrix get multiplied by input MACs during decryption, and these input MACs contain a product with Δ. To make this work, $\mathcal{S}_{\mathsf{Enc}}$ uses the MAC $\boldsymbol{X}$ it receives for $\boldsymbol{x}$, and substitutes $\Delta\boldsymbol{x}' = \Delta(\boldsymbol{x}' - \boldsymbol{x}) + \boldsymbol{X}$ into the equation. In more detail, $\mathcal{S}_{\mathsf{Enc}}$ rewrites the function $f(\Delta)(\boldsymbol{x}')$ in terms of $\boldsymbol{v} = \boldsymbol{x}' - \boldsymbol{x}$ and $\Delta \cdot \boldsymbol{v}$, adding in $y - f(\Delta)(\boldsymbol{x})$ (which is 0 at the correct Δ) to cancel out the term linear in Δ, then shifts this function to take $\overline{\boldsymbol{x}}' = \boldsymbol{x}' + \boldsymbol{m}$ instead of $\boldsymbol{v}$ and the MAC $\boldsymbol{X}' = \boldsymbol{X} + \Delta \cdot \boldsymbol{v}$ instead of $\Delta \cdot \boldsymbol{v}$.

$\underline{\mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, y, \boldsymbol{x}, f)}$:

$u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2} \leftarrow [\ell]$

let $\boldsymbol{F}' = \begin{bmatrix} \boldsymbol{F}'_{11} & \boldsymbol{F}'_{12} & F'_{13} \\ \boldsymbol{F}'_{21} & \boldsymbol{F}'_{22} & 0 \\ \boldsymbol{F}'_{31} & \boldsymbol{F}'_{32} & 0 \end{bmatrix}$ such that

$$f(\Delta)(\boldsymbol{x} + \boldsymbol{v}) - f(\Delta)(\boldsymbol{x}) + y = \begin{bmatrix} 1 \\ \boldsymbol{v}_1 \\ \Delta \cdot \boldsymbol{v}_1 \end{bmatrix}^{\mathsf{T}} \boldsymbol{F}' \begin{bmatrix} 1 \\ \boldsymbol{v}_2 \\ \Delta \cdot \boldsymbol{v}_2 \end{bmatrix}$$

$$\boldsymbol{F}^* = \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_1 & -\boldsymbol{X}_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_2 & -\boldsymbol{X}_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}$$

$$\text{return } \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & u_{1,1} \\ 0 & 1 & 0 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{F}^* \cdot \mathsf{f} & \mathsf{pk}_1 \\ \mathsf{pk}_2^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & u_{2,1} \\ 0 & 1 & 0 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}$$

Next, we present a hybrid argument for ciphertext simulation, starting from the simulation and ending at the real world.

Hybrid_1: This is the simulated distribution from Definition 10:

$$\boxed{\begin{array}{l}\text{for } \sigma \in [2], \begin{cases}\mathsf{pk}_\sigma & = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\rho_\sigma, \mathsf{pk}_w) \\ (\overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) & = \mathsf{Re}_{\mathsf{MAC}}(\rho_\sigma, (\boldsymbol{x}_\sigma, 0))\end{cases} \\ c \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, f(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f) \\ \text{return } c\end{array}}$$

with $\mathcal{S}_{\mathsf{Enc}}$ as defined above.

Hybrid_2: Let $\mathbf{C} = \begin{bmatrix}\overline{\mathbf{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0\end{bmatrix}$, and rewrite $\overline{\mathbf{C}}$ as

$$\begin{aligned}\overline{\mathbf{C}} &= \begin{bmatrix}1 & 0 & 0 & u_{1,1} \\ 0 & 1 & 0 & u_{1,2} \\ 0 & 0 & 1 & 0\end{bmatrix} \begin{bmatrix}\boldsymbol{F}^* \cdot \mathsf{f} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & u_{2,1} \\ 0 & 1 & 0 & u_{2,2} \\ 0 & 0 & 1 & 0\end{bmatrix}^\mathsf{T} \\ &= \boldsymbol{F}^* \cdot \mathsf{f} + \begin{bmatrix}u_{1,1} \\ u_{1,2} \\ 0\end{bmatrix} \mathsf{pk}_2^\mathsf{T} + \mathsf{pk}_1 \begin{bmatrix}u_{2,1} \\ u_{2,2} \\ 0\end{bmatrix}^\mathsf{T} \\ &= \boldsymbol{F}^* \cdot \mathsf{f} + \begin{bmatrix}u_{1,1} \cdot \mathsf{g} \\ u_{1,2} \cdot \mathsf{g} \\ 0\end{bmatrix} \begin{bmatrix}K_2 \\ \Delta \\ -1\end{bmatrix}^\mathsf{T} + \begin{bmatrix}K_1 \\ \Delta \\ -1\end{bmatrix} \begin{bmatrix}u_{2,1} \cdot \mathsf{g} \\ u_{2,2} \cdot \mathsf{g} \\ 0\end{bmatrix}^\mathsf{T}\end{aligned}$$

Hybrid_3: Write

$$\boldsymbol{F}^* = \begin{bmatrix}F^*_{11} & F^*_{12} & F^*_{13} \\ F^*_{21} & F^*_{22} & 0 \\ F^*_{31} & F^*_{32} & 0\end{bmatrix}.$$

The only place the $u_{i,j}$ are used now is by multiplication with g. Replace $u_{i,j} \cdot \mathsf{g}$ with $u_{i,j} \cdot \mathsf{g} + \hat{u}_{i,j} \cdot \mathsf{f}$, where $\hat{u}_{1,1} = F^*_{13}$, $\hat{u}_{1,2} = 0$, $\hat{u}_{2,1} = F^*_{31}$, and $\hat{u}_{2,2} = F^*_{32}$. This is indistinguishable by Lemma 6.

Hybrid_4: Rewrite the computation of $\overline{\boldsymbol{C}}$ in terms of a matrix $\hat{\boldsymbol{F}}$ as follows.

$$\overline{\mathbf{C}} = \underbrace{\left(\boldsymbol{F}^* + \begin{bmatrix}\hat{u}_{1,1} \\ \hat{u}_{1,2} \\ 0\end{bmatrix} \begin{bmatrix}K_2 \\ \Delta \\ -1\end{bmatrix}^\mathsf{T} + \begin{bmatrix}K_1 \\ \Delta \\ -1\end{bmatrix} \begin{bmatrix}\hat{u}_{2,1} \\ \hat{u}_{2,2} \\ 0\end{bmatrix}^\mathsf{T}\right)}_{\hat{\boldsymbol{F}}} \cdot \mathsf{f} + \begin{bmatrix}u_{1,1} \\ u_{1,2} \\ 0\end{bmatrix} \begin{bmatrix}K_2 \\ \Delta \\ -1\end{bmatrix}^\mathsf{T} \cdot \mathsf{g} + \begin{bmatrix}K_1 \\ \Delta \\ -1\end{bmatrix} \begin{bmatrix}u_{2,1} \\ u_{2,2} \\ 0\end{bmatrix}^\mathsf{T} \cdot \mathsf{g}$$

Next, use the definition of $\hat{u}_{i,j}$ to rewrite $\hat{\boldsymbol{F}}$ as:

$$\begin{aligned}\hat{\boldsymbol{F}} &= \begin{bmatrix}F^*_{11} & F^*_{12} & F^*_{13} \\ F^*_{21} & F^*_{22} & 0 \\ F^*_{31} & F^*_{32} & 0\end{bmatrix} + \begin{bmatrix}F^*_{13} \\ 0 \\ 0\end{bmatrix} \begin{bmatrix}K_2 \\ \Delta \\ -1\end{bmatrix}^\mathsf{T} + \begin{bmatrix}K_1 \\ \Delta \\ -1\end{bmatrix} \begin{bmatrix}F^*_{31} \\ F^*_{32} \\ 0\end{bmatrix}^\mathsf{T} \\ &= \begin{bmatrix}F^*_{11} + F^*_{31}K_1 + F^*_{13}K_2 & F^*_{12} + F^*_{32}K_1 + F^*_{13}\Delta & 0 \\ F^*_{21} + F^*_{31}\Delta & F^*_{22} + F^*_{32}\Delta & 0 \\ 0 & 0 & 0\end{bmatrix} \\ &= \begin{bmatrix}\overline{\boldsymbol{F}} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0\end{bmatrix}\end{aligned}$$

where

$$\begin{aligned}
\overline{\boldsymbol{F}} &= \begin{bmatrix} F^*_{11} + F^*_{31}K_1 + F^*_{13}K_2 & F^*_{12} + F^*_{32}K_1 + F^*_{13}\Delta \\ F^*_{21} + F^*_{31}\Delta & F^*_{22} + F^*_{32}\Delta \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & K_1 \\ 0 & 1 & \Delta \end{bmatrix} F^* \begin{bmatrix} 1 & 0 & K_2 \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T} \\
&= \begin{bmatrix} 1 & 0 & K_1 \\ 0 & 1 & \Delta \end{bmatrix} \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_1 & -\boldsymbol{X}_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_2 & -\boldsymbol{X}_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 & 0 & K_2 \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T} \\
&= \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_1 & K_1 - \boldsymbol{X}_1 \\ 0 & 1 & \Delta \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_2 & K_2 - \boldsymbol{X}_2 \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T}
\end{aligned}$$

Let $\boldsymbol{F}$ be the matrix representing $f(\Delta) \in \mathcal{F}_{\mathsf{pp}}$, i.e., $f(\boldsymbol{x}) = [1\ \boldsymbol{x}_1]\boldsymbol{F}[1\ \boldsymbol{x}_2]^\mathsf{T}$. We claim that

$$\overline{\boldsymbol{F}} = \begin{bmatrix} 1 & -\boldsymbol{m}_1 \\ 0 & 1 \end{bmatrix} \boldsymbol{F} \begin{bmatrix} 1 & -\boldsymbol{m}_2 \\ 0 & 1 \end{bmatrix}^\mathsf{T}$$

where $\boldsymbol{m} = \overline{\boldsymbol{x}} - \boldsymbol{x}$ is the input mask. To see this, we will evaluate on a masked input $\overline{\boldsymbol{x}}' = \boldsymbol{x}' + \boldsymbol{m}$:

$$\begin{aligned}
\begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_1 \end{bmatrix}^\mathsf{T} \overline{\boldsymbol{F}} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_2 \end{bmatrix} &= \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_1 & -\Delta\overline{\boldsymbol{x}}_1 \\ 0 & 1 & \Delta \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_2 & -\Delta\overline{\boldsymbol{x}}_2 \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & \overline{\boldsymbol{x}}'_1 - \overline{\boldsymbol{x}}_1 & \Delta \cdot (\overline{\boldsymbol{x}}'_1 - \overline{\boldsymbol{x}}_1) \\ 0 & 1 & \Delta \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & \overline{\boldsymbol{x}}'_2 - \overline{\boldsymbol{x}}_2 & \Delta \cdot (\overline{\boldsymbol{x}}'_2 - \overline{\boldsymbol{x}}_2) \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T} \\
&= \begin{bmatrix} 1 & \boldsymbol{x}'_1 - \boldsymbol{x}_1 & \Delta \cdot (\boldsymbol{x}'_1 - \boldsymbol{x}_1) \\ 0 & 1 & \Delta \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & \boldsymbol{x}'_2 - \boldsymbol{x}_2 & \Delta \cdot (\boldsymbol{x}'_2 - \boldsymbol{x}_2) \\ 0 & 1 & \Delta \end{bmatrix}^\mathsf{T} \\
&= f(\Delta)(\boldsymbol{x}') - f(\Delta)(\boldsymbol{x}) + y = f(\Delta)(\boldsymbol{x}'),
\end{aligned}$$

by definition of $\boldsymbol{F}'$. For $\boldsymbol{F}$ we get:

$$\begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 & -\boldsymbol{m}_1 \\ 0 & 1 \end{bmatrix} \boldsymbol{F} \begin{bmatrix} 1 & -\boldsymbol{m}_2 \\ 0 & 1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_1 - \boldsymbol{m}_1 \end{bmatrix}^\mathsf{T} \boldsymbol{F} \begin{bmatrix} 1 \\ \overline{\boldsymbol{x}}'_2 - \boldsymbol{m}_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \boldsymbol{x}'_1 \end{bmatrix}^\mathsf{T} \boldsymbol{F} \begin{bmatrix} 1 \\ \boldsymbol{x}'_2 \end{bmatrix} = f(\boldsymbol{x}'),$$

by definition of $\boldsymbol{F}$. Therefore, we change the experiment to compute

$$\hat{\boldsymbol{F}} = \begin{bmatrix} 1 & -\boldsymbol{m}_1 & -K_1 \\ 0 & 1 & -\rho_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} \\ \boldsymbol{0}^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} 1 & -\boldsymbol{m}_2 & -K_2 \\ 0 & 1 & -\rho_2 \\ 0 & 0 & 1 \end{bmatrix}^\mathsf{T},$$

and the result is identical to the previous hybrid.

We now compare with the real distribution:

$\overline{\rho}'_c \leftarrow \mathcal{G}_{\mathsf{Enc}}(1^\kappa)$
$\mathbf{C} = \mathsf{Rerand}(\overline{\rho}'_c \star \rho_c, \mathsf{Compile}(f(\Delta)))$
return $\mathbf{C}$

In the real distribution, we have

$$\mathbf{C} = \underbrace{\begin{bmatrix} 1 & -m_1 & -K_1 & u_{1,1} \\ 0 & 1 & -\rho_1 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R}_{\text{in-1}}} \cdot \begin{bmatrix} \boldsymbol{F} \cdot \mathsf{f} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 & -\mathsf{g} \\ \mathbf{0}^\mathsf{T} & -\mathsf{g} & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & -m_2 & -K_2 & u_{2,1} \\ 0 & 1 & -\rho_2 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T}}_{\boldsymbol{R}_{\text{in-2}}}$$

$$= \begin{bmatrix} 1 & 0 & 0 & u_{1,1} \\ 0 & 1 & 0 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{\boldsymbol{F}} \cdot \mathsf{f} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & u_{2,1} \\ 0 & 1 & 0 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T}$$

$$= \begin{bmatrix} \overline{\mathbf{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix}$$

where

$$\overline{\mathbf{C}} = \hat{\boldsymbol{F}} \cdot \mathsf{f} + \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ 0 \end{bmatrix} \mathsf{pk}_2^\mathsf{T} + \mathsf{pk}_1 \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ 0 \end{bmatrix}^\mathsf{T} = \hat{\boldsymbol{F}} \cdot \mathsf{f} + \begin{bmatrix} u_{1,1} \cdot \mathsf{g} \\ u_{1,2} \cdot \mathsf{g} \\ 0 \end{bmatrix} \begin{bmatrix} K_2 \\ \Delta \\ -1 \end{bmatrix}^\mathsf{T} + \begin{bmatrix} K_1 \\ \Delta \\ -1 \end{bmatrix} \begin{bmatrix} u_{2,1} \cdot \mathsf{g} \\ u_{2,2} \cdot \mathsf{g} \\ 0 \end{bmatrix}^\mathsf{T},$$

which matches the last hybrid. The only difference is that in the last hybrid $u_{i,j} \leftarrow [\ell]$, when in the real world the $u_{i,j}$ are shifted by the ciphertext randomnesses present in ρ_c. Since $u_{i,j}$ is only used for multiplication with $\mathsf{g} \in \mathbb{H}$, the ℓ-Bounded Keys property (Algorithm 1) gives that both distributions are indistinguishable from sampling $u_{i,j} \cdot \mathsf{g}$ uniformly from $\langle \mathsf{g} \rangle$. Therefore, the last hybrid and the real world are indistinguishable.

MAC Simulation. Define the simulator as follows:

$\underline{\mathcal{S}_{\mathsf{MAC}}(\boldsymbol{R}_\sigma, \mathsf{pk}, \overline{x}_\sigma)}$:
let $\begin{bmatrix} 1 & -m & -K & 0 \\ 0 & 1 & -\Delta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \overline{\boldsymbol{R}}_\sigma$
$K' \leftarrow \left[2^\kappa \sqrt{qB\ell}\right]$
$X_\sigma = K' + K + \Delta \cdot \overline{x}_\sigma$
$\mathsf{pk}_\sigma = [X_\sigma \cdot \mathsf{g} - \overline{x}_\sigma \cdot \mathsf{pk} \quad \mathsf{pk} \quad -\mathsf{g}]^\mathsf{T}$
return $(\mathsf{pk}_\sigma, X_\sigma)$

We now argue that the only difference between the simulated distribution from Definition 10

$\Delta = \mathsf{key}(\rho_\sigma)$
$\mathsf{pk} = \mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)$
$\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$
$(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
return $(\mathsf{pk}_\sigma, X_\sigma)$

and the real distribution

$\overline{\rho}'_\sigma \leftarrow \mathcal{G}_{\mathsf{MAC}}(\boldsymbol{m}'_\sigma)$
$\mathsf{pk}_\sigma = \mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}'_\sigma \star \Delta' \star \rho_\sigma, \mathsf{pk}_w)$
$X_\sigma = \mathsf{MAC}(\overline{\rho}'_\sigma \star \Delta' \star \rho_\sigma, \boldsymbol{x}_\sigma)$
return $(\mathsf{pk}_\sigma, X_\sigma)$

is an offset applied to K'. That is, if we replaced K' with $K' + \Delta' \cdot (m_\sigma + x_\sigma) - \Delta \cdot \boldsymbol{m}'_\sigma$ (where $m_\sigma = \mathsf{offset}(\rho_\sigma)$) in the simulated distribution, then it would be identical to the real distribution. In both distributions we have

$$\mathsf{pk}_\sigma^\mathsf{T} \begin{bmatrix} 1 \\ \overline{x}_\sigma \\ X_\sigma \end{bmatrix} = 0 \quad \text{and} \quad \mathsf{pk}_\sigma = [\overline{\mathsf{pk}_\sigma}\ \mathsf{pk}\ -\mathsf{g}]^\mathsf{T},$$

which fully defines pk_σ in terms of $\overline{x}_\sigma$ X_σ, pk, and g. Therefore, the only difference can be in the distribution of $\overline{x}_\sigma$ and X_σ. In the real world,

$$\begin{aligned}
\begin{bmatrix} 1 \\ \overline{x}_\sigma \\ X_\sigma \end{bmatrix} &= \begin{bmatrix} 1 & -m'_\sigma & -K'_\sigma \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-\mathsf{T}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\Delta' \\ 0 & 0 & 1 \end{bmatrix}^{-\mathsf{T}} \begin{bmatrix} 1 & -m_\sigma & -K_\sigma \\ 0 & 1 & -\Delta \\ 0 & 0 & 1 \end{bmatrix}^{-\mathsf{T}} \begin{bmatrix} 1 \\ x_\sigma \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ m'_\sigma & 1 & 0 \\ K'_\sigma & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \Delta' & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ m_\sigma & 1 & 0 \\ K_\sigma + m_\sigma \Delta & \Delta & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x_\sigma \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ m'_\sigma & 1 & 0 \\ K'_\sigma & \Delta' & 1 \end{bmatrix} \begin{bmatrix} 1 \\ m_\sigma + x_\sigma \\ K_\sigma + (m_\sigma + x_\sigma)\Delta \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ m'_\sigma + m_\sigma + x_\sigma \\ K'_\sigma + K_\sigma + (m_\sigma + x_\sigma)(\Delta + \Delta') \end{bmatrix},
\end{aligned}$$

so $\overline{x}_\sigma$ is exactly as in the simulation, while X_σ differs by

$$(\Delta + \Delta') \cdot (m_\sigma + x_\sigma) - \Delta \cdot (\boldsymbol{m}'_\sigma + m_\sigma + x_\sigma) = \Delta' \cdot (m_\sigma + x_\sigma) - \Delta \cdot \boldsymbol{m}'_\sigma,$$

as promised. By Equations (4) and (9), this shift to K' has statistical distance at most

$$\frac{|\Delta' \cdot (m_\sigma + x_\sigma) - \Delta \cdot \boldsymbol{m}'_\sigma|}{2^\kappa \sqrt{qB\ell}} \leq \frac{\ell B(2^\kappa T + 1) + \ell T 2^\kappa B}{2^\kappa \sqrt{qB\ell}} \leq (2 + 2^{-\kappa}) \sqrt{\frac{B\ell}{q}} \cdot T.$$

Plugging in $T = 2^{-\kappa-3}\sqrt{\frac{q}{B\ell}}$, this is at most

$$(2 + 2^{-\kappa}) \sqrt{\frac{B\ell}{q}} \cdot 2^{-\kappa-3} \sqrt{\frac{q}{B\ell}} \leq 2^{-\kappa-1},$$

which is negligible.

Privacy. We start by defining the simulator Sim, which is given the evaluation $f(\Delta)(\boldsymbol{x})$ must produce a fake encryption that agrees with it. Essentially, Sim will just encrypt the constant function.

$\underline{\mathsf{Sim}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, y)}$:

$u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2} \leftarrow [\ell]$

$$\text{return } \mathsf{C} = \begin{bmatrix} 1 & 0 & 0 & u_{1,1} \\ 0 & 1 & 0 & u_{1,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} y \cdot \mathsf{f} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{matrix} \mathsf{pk}_1 \\ \\ 0 \end{matrix} \right] \begin{bmatrix} 1 & 0 & 0 & u_{2,1} \\ 0 & 1 & 0 & u_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T}$$

We now present a hybrid argument for privacy, starting from the $\mathcal{S}_{\mathsf{Enc}}$ encryption and ending with the Sim simulation.

Hybrid_1: This is the $\mathcal{S}_{\mathsf{Enc}}$ distribution from Privacy (Definition 10):

$$\boxed{\begin{array}{l}\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)\\ \mathsf{pk} = \mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)\\ (\mathsf{state}, \mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, \boldsymbol{x}, y, f) \leftarrow \mathcal{A}(\mathsf{pk})\\ \text{abort if } \exists \sigma \in [2].\ (\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \notin \mathcal{W}\\ c \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, y, \boldsymbol{x}, f)\\ \text{return } (c, \mathsf{state})\end{array}}$$

with $\mathcal{S}_{\mathsf{Enc}}$ as defined for Ciphertext Simulation. Our goal is to swap out the call to $\mathcal{S}_{\mathsf{Enc}}$ for Sim, and leave the rest unchanged.

We analyze $\mathbf{C}$ by writing it in terms of its 3×3 submatrix $\overline{\mathbf{C}}$: $\mathbf{C} = \begin{bmatrix} \overline{\mathbf{C}} & \mathsf{pk}_1 \\ \mathsf{pk}_2^\mathsf{T} & 0 \end{bmatrix}$. Follow the argument for Ciphertext Simulation in rewriting $\overline{\mathbf{C}}$ to get

$$\overline{\mathbf{C}} = \boldsymbol{F}^* \cdot \mathsf{f} + \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ 0 \end{bmatrix} \mathsf{pk}_2^\mathsf{T} + \mathsf{pk}_1 \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ 0 \end{bmatrix}^\mathsf{T}.$$

If abort did not occur, we know $(\mathsf{pk}_\sigma, \overline{\boldsymbol{x}}_\sigma, \boldsymbol{X}_\sigma) \in \mathcal{W}$ for $\sigma \in [2]$. This implies $\overline{\mathsf{pk}}_\sigma = X_\sigma \cdot \mathsf{g} - \overline{x}_\sigma \cdot \mathsf{pk}$. Substitute it in to get

$$\begin{aligned}\overline{\mathbf{C}} &= \boldsymbol{F}^* \cdot \mathsf{f} + \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ 0 \end{bmatrix} \begin{bmatrix} X_2 \cdot \mathsf{g} - \overline{x}_2 \cdot \mathsf{pk} \\ \mathsf{pk} \\ -\mathsf{g} \end{bmatrix}^\mathsf{T} + \begin{bmatrix} X_1 \cdot \mathsf{g} - \overline{x}_1 \cdot \mathsf{pk} \\ \mathsf{pk} \\ -\mathsf{g} \end{bmatrix} \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ 0 \end{bmatrix}^\mathsf{T} \\ &= \boldsymbol{F}^* \cdot \mathsf{f} + \begin{bmatrix} u_{1,1} \cdot \mathsf{g} \\ u_{1,2} \cdot \mathsf{g} \\ 0 \end{bmatrix} \begin{bmatrix} X_2 \\ 0 \\ -1 \end{bmatrix}^\mathsf{T} + \begin{bmatrix} u_{1,1} \cdot \mathsf{pk} \\ u_{1,2} \cdot \mathsf{pk} \\ 0 \end{bmatrix} \begin{bmatrix} -\overline{x}_2 \\ 1 \\ 0 \end{bmatrix}^\mathsf{T} + \begin{bmatrix} X_1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} u_{2,1} \cdot \mathsf{g} \\ u_{2,2} \cdot \mathsf{g} \\ 0 \end{bmatrix}^\mathsf{T} + \begin{bmatrix} -\overline{x}_1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} u_{2,1} \cdot \mathsf{pk} \\ u_{2,2} \cdot \mathsf{pk} \\ 0 \end{bmatrix}^\mathsf{T}\end{aligned}$$

where $\mathsf{pk} = (\Delta' + \Delta) \cdot \mathsf{g}$ for $\Delta' \leftarrow [\ell]$.

Hybrid_2: Switch to $\mathsf{pk} = \Delta' \cdot \mathsf{g}$ for $\Delta' \leftarrow [\ell]$. Since Δ' is only used in pk, and $\mathsf{g} \in \mathbb{H}$, this follows from the ℓ-Bounded Keys property definition 5: both distributions sample pk indistinguishably from uniformly from $\langle \mathsf{g} \rangle$.

Hybrid_3: Notice that Δ' is only used to compute pk, and that all $u_{i,j}$ are only used in $\overline{\mathbf{C}}$. Therefore, we can use Lemma 6 to shift (some of) the $u_{i,j} \cdot \mathsf{g}$ and $u_{i,j} \cdot \mathsf{pk}$ products by a multiple of f:

$$\begin{aligned} u_{1,1} \cdot \mathsf{g} &\mapsto u_{1,1} \cdot \mathsf{g} + F^*_{13} \cdot \mathsf{f} \\ u_{2,1} \cdot \mathsf{g} &\mapsto u_{2,1} \cdot \mathsf{g} + F^*_{31} \cdot \mathsf{f} \\ u_{2,2} \cdot \mathsf{g} &\mapsto u_{2,2} \cdot \mathsf{g} + F^*_{32} \cdot \mathsf{f} \\ u_{1,1} \cdot \mathsf{pk} &\mapsto u_{1,1} \cdot \mathsf{pk} - F^*_{12} \cdot \mathsf{f} - \overline{x}_1 \cdot F^*_{22} \cdot \mathsf{f} - X_1 \cdot F^*_{32} \cdot \mathsf{f} \\ u_{2,1} \cdot \mathsf{pk} &\mapsto u_{2,1} \cdot \mathsf{pk} - F^*_{21} \cdot \mathsf{f} \\ u_{2,2} \cdot \mathsf{pk} &\mapsto u_{2,2} \cdot \mathsf{pk} - F^*_{22} \cdot \mathsf{f} \end{aligned}$$

Under this change, we get that the multiplies of pk and g stay the same, while the multiple of f becomes

$$\boldsymbol{F}^* \cdot \mathsf{f} \mapsto \widetilde{\boldsymbol{F}} \cdot \mathsf{f}$$

where

$$\begin{aligned}
\widetilde{F} &= \begin{bmatrix} F^*_{11} & F^*_{12} & F^*_{13} \\ F^*_{21} & F^*_{22} & 0 \\ F^*_{31} & F^*_{32} & 0 \end{bmatrix} + \begin{bmatrix} F^*_{13} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} X_2 \\ 0 \\ -1 \end{bmatrix}^\mathsf{T} - \begin{bmatrix} F^*_{12} + \overline{x}_1 \cdot F^*_{22} + X_1 \cdot F^*_{32} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} -\overline{x}_2 \\ 1 \\ 0 \end{bmatrix}^\mathsf{T} \\
&\quad + \begin{bmatrix} X_1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} F^*_{31} \\ F^*_{32} \\ 0 \end{bmatrix}^\mathsf{T} - \begin{bmatrix} -\overline{x}_1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} F^*_{21} \\ F^*_{22} \\ 0 \end{bmatrix}^\mathsf{T} \\
&= \begin{bmatrix} y^* & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
y^* &= F^*_{11} + \overline{x}_1 F^*_{21} + X_2 \cdot F^*_{13} + X_1 \cdot F^*_{31} + \overline{x}_2 (F^*_{12} + \overline{x}_1 \cdot F^*_{22} + X_1 \cdot F^*_{32}) \\
&= \begin{bmatrix} 1 \\ \overline{x}_1 \\ X_1 \end{bmatrix}^\mathsf{T} F^* \begin{bmatrix} 1 \\ \overline{x}_2 \\ X_2 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ \overline{x}_1 \\ X_1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_1 & -\boldsymbol{X}_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{F}' \begin{bmatrix} 1 & -\overline{\boldsymbol{x}}_2 & -\boldsymbol{X}_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 \\ \overline{x}_2 \\ X_2 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^\mathsf{T} \boldsymbol{F}' \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = f(\Delta)(\boldsymbol{x}) - f(\Delta)(\boldsymbol{x}) + y = y,
\end{aligned}$$

by definition of $\boldsymbol{F}'$.

Hybrid_4: Undo the change in Hybrid_2, i.e., use ℓ-Bounded Keys to switch back to $\mathsf{pk} = (\Delta' + \Delta) \cdot \mathsf{g}$ for $\Delta' \leftarrow [\ell]$. $\boldsymbol{C}$ is now generated exactly as in Sim. This completes the hybrid proof.

□

D.2 Proof of Lemma 9

Proof. We use the following lemma to apply the masking property despite issues with non-commutativity.

Lemma 22. *For all $\rho^*_\sigma \in \mathsf{R}_{\mathsf{MAC},\sigma}$ and $\overline{\rho}_\sigma \in \overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$ where $2n(\rho^*_\sigma) + n(\overline{\rho}_\sigma) \le \beta$,*

$$\{\overline{\rho'}_\sigma \star \rho^*_\sigma \star \overline{\rho}_\sigma\}_{\overline{\rho'}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} \overset{\mathrm{s}}{\approx} \{\overline{\rho'}_\sigma \star \rho^*_\sigma\}_{\overline{\rho'}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))}$$

And for all $\rho^*_c \in \mathsf{R}$ and $\overline{\rho}_c \in \overline{\mathsf{R}}$ where $2n(\rho^*_c) + n(\overline{\rho}_c) \le \beta$,

$$\{\overline{\rho'}_c \star \rho^*_c \star \overline{\rho}_c\}_{\overline{\rho'}_c \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)} \overset{\mathrm{s}}{\approx} \{\overline{\rho'}_c \star \rho^*_c\}_{\overline{\rho'}_c \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta))}.$$

Proof. For the first statement, we have

$$\overline{\rho'}_\sigma \star \rho^*_\sigma \star \overline{\rho}_\sigma = \overline{\rho'}_\sigma \star (\rho^*_\sigma \star \overline{\rho}_\sigma \star (\rho^*_\sigma)^{-1}) \star \rho^*_\sigma,$$

so the statement will hold if we can apply masking to $\rho^*_\sigma \star \overline{\rho}_\sigma \star (\rho^*_\sigma)^{-1}$. It is in $\overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$ because $\overline{\mathsf{R}}_{\mathsf{MAC},\sigma}$ is a normal subgroup of $\mathsf{R}_{\mathsf{MAC},\sigma}$, and it has G-seminorm at most $n(\rho^*_\sigma) + n(\overline{\rho}_\sigma) + n((\rho^*_\sigma)^{-1})$ by subadditivity, so we can apply masking. Similarly, for the second statement

$$\overline{\rho'}_c \star \rho^*_c \star \overline{\rho}_c = \overline{\rho'}_c \star (\rho^*_c \star \overline{\rho}_c \star (\rho^*_c)^{-1}) \star \rho^*_c,$$

and again we can apply masking to $\rho^*_c \star \overline{\rho}_c \star (\rho^*_c)^{-1}$ because $\overline{\mathsf{R}}$ is a normal subgroup of R, and the G-seminorm is bounded by subadditivity. □

With this lemma out of the way, we prove the statement using a hybrid proof from the real distribution $\{\rho'_c \star \rho_c\}_{\rho'_c \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa,\beta)}$ to the random distribution $\{\rho'_c\}_{\rho'_c \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa,\beta))}$.

Hybrid_1: This is the real distribution. The sample is

$$\overline{\rho'}_c \star \overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta' \star \overline{\rho}_c \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta,$$

where $\Delta \leftarrow \mathcal{G}^*_{\mathsf{pk}}(1^\kappa,\beta)$, $\overline{\rho}_\sigma \leftarrow \mathcal{G}^*_{\mathsf{MAC}}(1^\kappa, 2(\beta_{\mathrm{mask}}+1)\beta)$ for $\sigma \in [2]$, and $\overline{\rho}_c \leftarrow \mathcal{G}^*_{\mathsf{Enc}}(1^\kappa, (8\beta^2_{\mathrm{mask}} + 10\beta_{\mathrm{mask}} + 2)\cdot\beta)$.

Hybrid_2: Remove $\overline{\rho}_c$, i.e., change the sample to

$$\overline{\rho'}_c \star \overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta' \star \overline{\rho}_1 \star \overline{\rho}_2 \star \Delta.$$

This is statistically indistinguishable by Lemma 22, as $n(\overline{\rho}_c) = n(\rho_c \star (\overline{\rho}_1 \star \overline{\rho}_2 \star \Delta)^{-1}) \le 2\beta$ and so

$$\begin{aligned} 2n(\overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta') + n(\overline{\rho}_c) &\le 2(2\cdot\beta_{\mathrm{mask}}2(\beta_{\mathrm{mask}}+1)\beta + \beta_{\mathrm{mask}}\beta) + 2\beta \\ &= (8\beta^2_{\mathrm{mask}} + 10\beta_{\mathrm{mask}} + 2)\beta \end{aligned}$$

Hybrid_3: Remove $\overline{\rho}_1$, so the sample becomes

$$\overline{\rho'}_c \star \overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta' \star \overline{\rho}_2 \star \Delta.$$

This is statistically indistinguishable by Lemma 22, as $n(\overline{\rho}_1) = n(\mathsf{in}_1(\rho_c)(\Delta)^{-1}) \le 2\beta$ and so $2n(\Delta') + n(\overline{\rho}_1) \le 2\beta_{\mathrm{mask}}\beta + 2\beta$. (Note that we do not need to count $\overline{\rho'}_2$ in the norm bound, because it commutes with $\overline{\rho'}_1$.)

Hybrid_4: Remove $\overline{\rho}_2$, so the sample becomes

$$\overline{\rho'}_c \star \overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta' \star \Delta.$$

This is statistically indistinguishable similarly to the last hybrid.

Hybrid_5: Remove Δ, so the sample becomes

$$\overline{\rho'}_c \star \overline{\rho'}_1 \star \overline{\rho'}_2 \star \Delta'.$$

This is statistically indistinguishable by masking, as $n(\Delta') \le n(\rho_c) \le \beta$.

□

E Proof of Theorem 14

Proof. We show that RGS in Algorithms 4 to 6 satisfies Definition 3 as follows:

Projective Encoding.

RGS satisfies projective encoding (Definition 2) since both the input encoding information $\boldsymbol{e}$ output by Gb and the transformations π_{in} are vectors of n elements and the algorithm En works as follows: for input $\boldsymbol{x} \in \mathcal{X}^{\mathsf{n}}$, $\mathsf{En}(\boldsymbol{e}, \boldsymbol{x})$ outputs $\boldsymbol{X} = \{(\overline{\boldsymbol{x}}_\sigma, X_\sigma)\}_{\sigma \in [\mathsf{n}]}$ where each $(\overline{\boldsymbol{x}}_\sigma, X_\sigma) \in \boldsymbol{X}$ is a function of $\boldsymbol{e}$ and only $x_\sigma \in \boldsymbol{x}$. Further $\pi_{\mathsf{in}} = \{\rho_\sigma\}_{\sigma \in [\mathsf{n}]}$ is such that each ρ_σ maps $(\overline{\boldsymbol{x}}_\sigma, X_\sigma)$ to the input encoding for input wire σ corresponding to the rerandomized garbling which is of the same form.

Structure on Gb.

As required in Definition 3, the garbling algorithm Gb has the structure,

$$\begin{array}{|l|}\hline \underline{\mathsf{Gb}(\mathbf{C})}: \\ \quad \rho \leftarrow \mathsf{Gen}(1^\kappa) \\ \quad (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}}) = \mathsf{Compile}(\mathbf{C}) \\ \quad (\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}') = \mathsf{Upd}(\rho, (\boldsymbol{F}_{\mathsf{pub}}, \boldsymbol{e}_{\mathsf{pub}}, \boldsymbol{d}_{\mathsf{pub}})) \\ \quad \text{return } (\boldsymbol{F}', \boldsymbol{e}', \boldsymbol{d}') \\ \hline \end{array}$$

where Compile, Upd and Rerand (used internally within Upd) are deterministic algorithms and Gen is used to sample garbling randomness from R_{re}.

Group Structure and Group Action.

For a circuit $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$ with $\mathsf{Q}_\times$ multiplication gates, the group of garbling randomness R_{re} contains all elements of the following form:

$$(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}_\sigma\}_{\sigma \in [\mathsf{n}]})$$

where,

- $\forall \sigma \in [\mathsf{n}]$, $\overline{\rho}_\sigma \in \mathsf{UGS}.\overline{\mathsf{R}}_{\mathsf{MAC}}$ belongs in the normal subgroup of $\mathsf{R}_{\mathsf{MAC}}$ that only shifts the wire value and MAC; isomorphic to $\langle \mathbb{Z}^2, + \rangle$.
- $\forall g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$, $\overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g} \in \mathsf{UGS}.\overline{\mathsf{R}}$ belongs in the normal subgroup of only encryption randomness, and $\overline{\rho}_\sigma \in \mathsf{UGS}.\overline{\mathsf{R}}_{\mathsf{MAC}}$ belongs in the normal subgroup of $\mathsf{R}_{\mathsf{MAC}}$ that only shifts the output wire value and MAC; both of which are isomorphic to $\langle \mathbb{Z}^2, + \rangle$.
- $\Delta \in \mathsf{UGS}.\mathsf{R}_{\mathsf{pk}}$ is in the subgroup transforming only the global key; isomorphic to $\langle \mathbb{Z}, + \rangle$.

It therefore follows from the group structure induced in UGS (definition 8) that the set of all elements of the above form also form a group with the group operator $\star$ defined in Algorithm 6. That is, identity, inverse, closure and associativity with respect to $\star$ are satisfied:

Identity. Let 1 be the identity element of the group UGS.R. Then the identity element of the group R_{re} is,

$$1_{\mathsf{Rerand}} = (1, \{1, 1, 1\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{1\}_{\sigma \in [\mathsf{n}]})$$

It follows from the description of the group operator $\star$ in Algorithm 6 that for any $\rho \in \mathsf{R}_{\mathsf{re}}$, $1_{\mathsf{Rerand}} \star \rho = \rho \star 1_{\mathsf{Rerand}} = \rho$.

Group Action. We must show that algorithm Upd described in Algorithms 4 and 5 is a group action. By inspection, $\mathsf{Upd}(1_{\mathsf{Rerand}}, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$, where 1_{Rerand} is the identity element of R_{re}. It remains to show that Upd is compatible with the group operator $\star$ in Algorithm 6, i.e., that $(\boldsymbol{F}_1, \boldsymbol{e}_1, \boldsymbol{d}_1) = \mathsf{Upd}(\rho^2 \star \rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = \mathsf{Upd}(\rho^2, \mathsf{Upd}(\rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))) = (\boldsymbol{F}_2, \boldsymbol{e}_2, \boldsymbol{d}_2)$.

Parsing the first output $\boldsymbol{F}_i$ of Upd, we have

$$\boldsymbol{F}_i = (\mathsf{pk}'_i, \{(c''_{1,g,i}, c''_{2,g,i}, \mathsf{pk}'_{\sigma,i})\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\mathsf{pk}'_{\sigma,i}\}_{\sigma \in [\mathsf{n}]})$$

Also parse the inputs ρ^1, ρ^2 and output $\rho' = \rho^2 \star \rho^1$ of the group operator $\star$:

$$\rho^i = (\Delta^i, \{\overline{\rho}^i_\sigma, \overline{\rho}^i_{c1,g}, \overline{\rho}^i_{c2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}^i_\sigma\}_{\sigma \in [\mathsf{n}]})$$
$$\rho' = (\Delta', \{\overline{\rho}'_\sigma, \overline{\rho}'_{c1,g}, \overline{\rho}'_{c2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}'_\sigma\}_{\sigma \in [\mathsf{n}]})$$

Then

$$\mathsf{pk}'_1 = \mathsf{UGS.Re}_{\mathsf{pk}}(\Delta', \mathsf{pk}) = \mathsf{UGS.Re}_{\mathsf{pk}}(\Delta^2 \star \Delta^1, \mathsf{pk}) = \mathsf{UGS.Re}_{\mathsf{pk}}(\Delta^2, \mathsf{UGS.Re}_{\mathsf{pk}}(\Delta^1, \mathsf{pk})) = \mathsf{pk}'_2,$$

because $\mathsf{UGS.Re}_{\mathsf{pk}}$ is a group action. Similarly, the wire public keys match: $\mathsf{pk}'_{\sigma,1} = \mathsf{pk}'_{\sigma,2}$ because, for inputs and multiplication gate outputs, $\mathsf{UGS.Re}_{\mathsf{pk\text{-}wire}}$ is a group action and for these wires $\overline{\rho}'_\sigma \star \Delta' = \overline{\rho}^2_\sigma \star \Delta^2 \star \overline{\rho}^1_\sigma \star \Delta^1$, and for addition gate outputs, they are defined deterministically in terms of previous wire's public keys using $+$.

Now consider each multiplication gate. Let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$ be the indices of the input and output wires For the $c''_{1,g,i}$ ciphertexts, letting $\rho^i_{j,g} = \overline{\rho}^i_{c_j,g} \star \overline{\rho}^i_{\sigma_1} \star \overline{\rho}^i_{\sigma_2} \star \Delta^i$ for $i \in \{1, 2, \prime\}$, we have

$$\begin{aligned}
c''_{1,g,1} &= \mathsf{UGS.Rerand}(\rho'_{c_1,g}, c_{1,g} + \mathsf{offset}(\overline{\rho}'_\sigma) \cdot c^*_1) \\
&= \mathsf{UGS.Rerand}(\rho^2_{c_1,g} \star \rho^1_{c_1,g}, c_{1,g} + (\mathsf{offset}(\overline{\rho}^1_\sigma) + \mathsf{offset}(\overline{\rho}^2_\sigma)) \cdot c^*_1) \\
&= \mathsf{UGS.Rerand}(\rho^2_{c_1,g}, \mathsf{UGS.Rerand}(\rho^1_{c_1,g}, c_{1,g} + (\mathsf{offset}(\overline{\rho}^1_\sigma) + \mathsf{offset}(\overline{\rho}^2_\sigma)) \cdot c^*_1)) \\
&= \mathsf{UGS.Rerand}(\rho^2_{c_1,g}, \mathsf{UGS.Rerand}(\rho^1_{c_1,g}, c_{1,g} + \mathsf{offset}(\overline{\rho}^1_\sigma) \cdot c^*_1) + \mathsf{offset}(\overline{\rho}^2_\sigma) \cdot c^*_1) \\
&= c''_{1,g,2},
\end{aligned}$$

where c^*_1 by abuse of notation represents the ciphertext given by replacing the public keys in the output $(\mathsf{pk}_w, \mathsf{pk}_w, \bar{c}_1)$ of $\mathsf{Compile}(1)$ with the public keys of the other operand of addition, so that addition is defined; this notation is well defined because of the property of updatable garbling that Rerand only changes the public keys when run on $\mathsf{Compile}(1)$. In the first step, we used the Rerand algorithm (including the definition of f_{offset}). In the second, the computation of the ciphertext rerandomization values $\rho'_{c_1,g}$ and of the wire rerandomization values $\overline{\rho}'_\sigma$ in the $\star$ algorithm, and that offset is a homomorphism such that the global key is in its kernel. In the third, we used that $\mathsf{UGS.Rerand}$ is a group action. In the fourth, we use linearity of $\mathsf{UGS.Rerand}$. Finally, we matched the result to the computation in the Rerand algorithm.

For the $c''_{2,g,i}$ ciphertexts, the argument is similar, but has an extra complication because the rerandomization of the global key affects both the input and the output of the bilinear operation it represents. First, we have to define notation for what the affine function f_{MAC} does, which is defined based on $\mathsf{UGS.MAC}$. Because $\mathsf{UGS.MAC}$ is required to be a affine function in x, we can write

$$\mathsf{UGS.MAC}(\rho_\sigma, x) = a(\rho_\sigma) \cdot x + b(\rho_\sigma)$$

for some functions a and b. Because $\mathsf{Re}_{\mathsf{MAC}}$ is a group action, we have

$$\begin{aligned}
\mathsf{UGS.MAC}(\rho^2_\sigma \star \rho^1_\sigma, x) &= \mathsf{MAC}(\rho^1_\sigma, x) + \mathsf{UGS.MAC}(\rho^2_\sigma, x + \mathsf{offset}(\rho^1_\sigma)) \\
a(\rho^2_\sigma \star \rho^1_\sigma) &= a(\rho^1_\sigma) + a(\rho^2_\sigma) \\
b(\rho^2_\sigma \star \rho^1_\sigma) &= b(\rho^1_\sigma) + b(\rho^2_\sigma) + a(\rho^2_\sigma)\, \mathsf{offset}(\rho^1_\sigma).
\end{aligned}$$

Starting from the composed randomness ρ' and using the definitions of Rerand and $\star$, we have

$$\begin{aligned} c''_{2,g,1} &= \mathsf{UGS.Rerand}(\rho'_{c_{2,g}},\ c_{2,g} + a(\rho'_\sigma) \cdot c_{1,g} + b(\rho'_\sigma) \cdot c_1^*) \\ &= \mathsf{UGS.Rerand}(\rho^2_{c_{2,g}} \star (\overline{\rho}^1_{c_{1,g}})^{a(\rho^2_\sigma)} \star \rho^1_{c_{2,g}},\ c_{2,g} + a(\rho'_\sigma) \cdot c_{1,g} + b(\rho'_\sigma) \cdot c_1^*) \\ &= \mathsf{UGS.Rerand}(\rho^2_{c_{2,g}}, u_g), \end{aligned}$$

where

$$u_g = \mathsf{UGS.Rerand}((\overline{\rho}^1_{c_{1,g}})^{a(\rho^2_\sigma)} \star \rho^1_{c_{2,g}},\ c_{2,g} + a(\rho^1_\sigma) \cdot c_{1,g} + a(\rho^2_\sigma) \cdot c_{1,g} + b(\rho'_\sigma) \cdot c_1^*),$$

because $\mathsf{UGS.Rerand}$ is a group action. Next, we compute this intermediate value u_g. Using the linearity of $\mathsf{UGS.Rerand}$ twice (which works because $\mathsf{in}_\sigma(\rho^1_{c_{1,g}}) = \mathsf{in}_\sigma(\rho^1_{c_{2,g}})$), along with the calculation of $b(\rho'_\sigma)$ above, we get

$$\begin{aligned} u_g &= \mathsf{UGS.Rerand}(\rho^1_{c_{2,g}},\ c_{2,g} + a(\rho^1_\sigma) \cdot c_{1,g} + b(\rho'_\sigma) \cdot c_1^*) + a(\rho^2_\sigma) \cdot \mathsf{UGS.Rerand}(\rho^1_{c_{1,g}}, c_{1,g}) \\ &= \mathsf{UGS.Rerand}(\rho^1_{c_{2,g}},\ c_{2,g} + a(\rho^1_\sigma) \cdot c_{1,g} + b(\rho^1_\sigma) \cdot c_1^*) + a(\rho^2_\sigma) \cdot \hat{c}_{1,g} + b(\rho^2_\sigma) \cdot c_1^* \end{aligned}$$

where

$$\hat{c}_{1,g} = \mathsf{UGS.Rerand}(\rho^1_{c_{1,g}}, c_{1,g} + \mathsf{offset}(\overline{\rho}^1_\sigma) \cdot c_1^*)$$

is the intermediate result of rerandomizing $c_{1,g}$ once, so

$$\mathsf{UGS.Rerand}(\rho^2_{c_{2,g}}, u_g) = c''_{2,g,2}.$$

We've now shown that group action compatibility holds for F. For d, the action is just adding offset output by offset, which is a homomorphism, so compatibility holds. For e, the action is the UGS's wire rerandomization $\star$, so compatibility holds by associativity.

Associativity. It may seem that we have to do another detailed analysis of $\star$, similarly to the analysis for group action compatibility. However, there is a trick that lets us reuse the analysis above. The idea is that if the group action is faithful[8] then group action compatibility implies associativity. That is, group action compatibility says that theres a homomorphism from the magma defined by $\star$ into the group of invertible functions on $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$, and Upd being faithful lets you bring associativity back from the group of invertible functions to $\star$. In other words,

$$\begin{aligned} \mathsf{Upd}(\rho^3 \star (\rho^2 \star \rho^1), (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) &= \mathsf{Upd}(\rho^3, \mathsf{Upd}(\rho^2 \star \rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))) \\ &= \mathsf{Upd}(\rho^3, \mathsf{Upd}(\rho^2, \mathsf{Upd}(\rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})))) \\ &= \mathsf{Upd}(\rho^3 \star \rho^2, \mathsf{Upd}(\rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))) \\ &= \mathsf{Upd}((\rho^3 \star \rho^2) \star \rho^1, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) \end{aligned}$$

so if the action is faithful then $\rho^3 \star (\rho^2 \star \rho^1) = (\rho^3 \star \rho^2) \star \rho^1$.

Unfortunately, we don't know if the action is faithful. However, we can always modify the UGS group actions to make it faithful, without changing the $\star$ operations, which still lets us infer the associativity of $\star$. The modification to the UGS scheme is to make the public keys and ciphertexts always include the rerandomization values (Δ for the global public key, ρ_σ for the wire key, and ρ_c for a ciphertext) as well as their original information. I.e., each ciphertext will include a rerandomization group element $\rho_c \in \mathsf{R}$, which will start at the identity, $\star$ will be applied to it with each rerandomization, and just the $\overline{\rho}_c$ part will be multiplied when two ciphertexts get added. This preserves all the group action and linearity properties, so the argument for group action compatibility above will go through. And the modified scheme is faithful because the rerandomization value must be preserved in the output if the initial rerandomization value is the identity.

With this modified action $\star$ is associative, and the group action is the same as with the original UGS scheme, so we conclude that the original $\star$ is associative.

[8] Upd is faithful if for all ρ, ρ', $\mathsf{Upd}(\rho, (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})) = \mathsf{Upd}(\rho', (\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d}))$ for *all* $(\boldsymbol{F}, \boldsymbol{e}, \boldsymbol{d})$ implies $\rho = \rho'$.

Inverse. For each element $\rho^* = (\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}_\sigma\}_{\sigma\in[\mathsf{n}]}) \in \mathsf{R}_{\mathsf{re}}$, its inverse ρ^{*-1} is an element $(\Delta', \{\overline{\rho}'_\sigma, \overline{\rho}'_{c_1,g}, \overline{\rho}'_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}'_\sigma\}_{\sigma\in[\mathsf{n}]})$ where,

$$\begin{aligned}
&&\Delta' &= \Delta^{-1} \\
\forall \sigma \in [\mathsf{n}], && \overline{\rho}'_\sigma \star \Delta' &= (\overline{\rho}_\sigma \star \Delta)^{-1} \\
\forall g \in [\mathsf{Q}], \text{ if } \mathsf{op}_g = +, && \overline{\rho}'_\sigma &= \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \\
\text{if } \mathsf{op}_g = \times, && \overline{\rho}'_\sigma \star \Delta' &= (\overline{\rho}_\sigma \star \Delta)^{-1} \\
&& \overline{\rho}'_{c_1,g} \star \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \star \Delta' &= (\overline{\rho}_{c_1,g} \star \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)^{-1} \\
&& \overline{\rho}'_{c_2,g} \star \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \star \Delta' &= \left((\overline{\rho}_{c_1,g})^{a(\rho'_\sigma)} \star \overline{\rho}_{c_2,g} \star \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta\right)^{-1}
\end{aligned} \tag{10}$$

where $a(\rho'_\sigma) = \mathsf{UGS.MAC}(\rho'_\sigma, 1) - \mathsf{UGS.MAC}(\rho'_\sigma, 0)$ is defined the same as in the group action property. That is, compute each RHS in order, and decompose the result using the semidirect product structure (as guaranteed by Definition 8). Since $\mathsf{in}_1, \mathsf{in}_2, \mathsf{key}$ are homomorphisms, we have that the different equations are all consistent, i.e., the Δ', $\overline{\rho}'_{\sigma_1}$, and $\overline{\rho}'_{\sigma_2}$ elements will all match. Plugging these values into Algorithm 6 immediately shows that this procedure gives an inverse group element to ρ^*.

Invertible sampling.

The randomness sampler Gen described in Algorithm 4 has invertible sampling as its output is a tuple of elements, all the of which is sampled using the algorithms $\mathsf{UGS}.\mathcal{G}_{\mathsf{pk}}$, $\mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$, $\mathsf{UGS}.\mathcal{G}_{\mathsf{MAC}}$, and $\mathsf{UGS}.\mathcal{G}_{\mathsf{Enc}}$, all of which have invertible sampling within UGS (Definition 8). That is, if the output of the inverse sampler Sim for RGS is not statistically closely distributed to the output of Gen, this would imply the existence of at least one of the samplers in UGS that is also not statistically close. This yields a contradiction.

G-seminorm.

In addition to the free-addition norm, we need to keep track of another possible blow up, this time caused by $\overline{\rho}^i_{c_1,g}$ getting added into $\overline{\rho}^i_{c_2,g}$ according to MAC's linear coefficient; we therefore use Affine Boundedness to upper bound MAC's linear coefficient by β_{MAC}. Using the free-addition norm α and the MAC bound β_{MAC}, we define the n to be the maximum of three terms:

$$n(\rho) = \max \left(\begin{array}{l} \max\limits_{\substack{\sigma\in[\mathsf{n}+\mathsf{Q}] \\ \sigma\in[\mathsf{n}] \vee \mathsf{op}_{\sigma-\mathsf{n}}=\times}} n(\rho_\sigma), \\ \max\limits_{\substack{g\in[\mathsf{Q}] \\ \mathsf{op}_g=\times}} \dfrac{n(\rho_{c_1,g})}{4\alpha+3}, \\ \max\limits_{\substack{g\in[\mathsf{Q}] \\ \mathsf{op}_g=\times}} \sqrt{\dfrac{n\left((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2 \rfloor} \star \rho_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}} \end{array} \right)$$

The first term bounds the wire-rerandomization (and so also the global key), the second bounds the c_1 ciphertexts for each gate (which decrypt to the masked output wire value), and the third bounds the c_2 ciphertexts for each gate (which decrypt to the MAC of the output wire value). The latter two terms include inverse factors of α, to cancel out the free-addition blow up in $\overline{\rho}^i_{c_1,g}$ and $\overline{\rho}^i_{c_2,g}$, and the last also includes an inverse factor of β_{MAC}, to cancel out the blow up from the linear coefficient of MAC.

All three terms trivially satisfy nonnegativity and identity, and for the first two terms symmetry follows from symmetry of the G-seminorm in the underlying UGS groups. To show symmetry for the third term, calculate the inverse ρ' of ρ using Equation (10), and use symmetry

of the underlying G-seminorm. However, we first need to derive an identity to take care of a commutativity issue:

$$\begin{aligned}(\rho_{c_2,g})^{-1} \star (\overline{\rho}_{c_1,g})^{-1} \star \rho_{c_2,g} &= (\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)^{-1} \star (\overline{\rho}_{c_1,g})^{-1} \star (\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta) \\ &= (\overline{\rho}_{c_1,g} \star \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)^{-1} \star (\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta) \\ &= (\rho_{c_1,g})^{-1} \star ((\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)^{-1})^{-1} \\ &= \rho'_{c_1,g} \star (\overline{\rho'}_{\sigma_1} \star \overline{\rho'}_{\sigma_2} \star \Delta')^{-1} \\ &= \overline{\rho'}_{c_1,g},\end{aligned}$$

where we first used that $\overline{\mathsf{R}}$ is a commutative normal subgroup, and then the inverse operation for $\rho_{c_1,g}$ defined above in Equation (10). Then from Equation (10), we have

$$\rho'_{c_2,g} = \left((\overline{\rho}_{c_1,g})^{a(\rho'_\sigma)} \star \rho_{c_2,g}\right)^{-1} = \left((\overline{\rho}_{c_1,g})^{-a(\rho_\sigma)} \star \rho_{c_2,g}\right)^{-1} = (\rho_{c_2,g})^{-1}(\overline{\rho}_{c_1,g})^{a(\rho_\sigma)},$$

so

$$\begin{aligned}n\left((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2 \rfloor} \star \rho_{c_2,g}\right) &= n\left((\rho_{c_2,g})^{-1} \star (\overline{\rho}_{c_1,g})^{\lfloor a(\rho_\sigma)/2 \rfloor}\right) \\ &= n\left((\rho_{c_2,g})^{-1} \star (\overline{\rho}_{c_1,g})^{\lfloor a(\rho_\sigma)/2 \rfloor - a(\rho_\sigma)} \star (\overline{\rho}_{c_1,g})^{a(\rho_\sigma)}\right) \\ &= n\left((\overline{\rho'}_{c_1,g})^{a(\rho_\sigma) - \lfloor a(\rho_\sigma)/2 \rfloor} \star (\rho_{c_2,g})^{-1} \star (\overline{\rho}_{c_1,g})^{a(\rho_\sigma)}\right) \\ &= n\left((\overline{\rho'}_{c_1,g})^{\lceil a(\rho_\sigma)/2 \rceil} \star \rho'_{c_2,g}\right) \\ &= n\left((\overline{\rho'}_{c_1,g})^{\lceil -a(\rho'_\sigma)/2 \rceil} \star \rho'_{c_2,g}\right) \\ &= n\left((\overline{\rho'}_{c_1,g})^{-\lfloor a(\rho'_\sigma)/2 \rfloor} \star \rho'_{c_2,g}\right).\end{aligned}$$

For subadditivity, note that multiplication for the ρ_σ and $\rho_{c_1,g}$ is just multiplication in the underlying UGS group, so for the first two terms subadditivity is trivially implied by the UGS G-seminorm's subadditivity. For the third term, we follow a similar argument to subadditivity square-root terms in the UGS scheme's G-seminorm. That is, we have $\rho = \rho^2 \star \rho^1$ and we want to show that

$$n(\rho^2) + n(\rho^1) \geq \sqrt{\frac{n\left((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2 \rfloor} \star \rho_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}},$$

and do so by proving the square of both sides:

$$\begin{aligned}&\left(\max\left(n(\rho^2_\sigma), \frac{n(\rho^2_{c_1,g})}{4\alpha+3}, \sqrt{\frac{n\left((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2 \rfloor} \star \rho^2_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}}\right)\right. \\ &\left. + \max\left(n(\rho^1_\sigma), \frac{n(\rho^1_{c_1,g})}{4\alpha+3}, \sqrt{\frac{n\left((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2 \rfloor} \star \rho^1_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}}\right)\right)^2 \\ &\geq \frac{n\left((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2 \rfloor} \star \rho^2_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} + \frac{n\left((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2 \rfloor} \star \rho^1_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\ &+ n(\rho^1_\sigma) \cdot \frac{n(\rho^2_{c_1,g})}{4\alpha+3} + \frac{n(\rho^1_{c_1,g})}{4\alpha+3} \cdot n(\rho^2_\sigma) \\ &= \frac{n\left((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2 \rfloor} \star \rho^2_{c_2,g}\right) + n\left((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2 \rfloor} \star \rho^1_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\ &+ \frac{2\beta_{\mathsf{MAC}} n(\rho^1_\sigma) n(\rho^2_{c_1,g}) + 2\beta_{\mathsf{MAC}} n(\rho^1_{c_1,g}) n(\rho^2_\sigma)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\ &\geq \frac{n\left((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2 \rfloor} \star \rho^2_{c_2,g}\right) + n\left((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2 \rfloor} \star \rho^1_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}\end{aligned}$$

$$
\begin{aligned}
&+ \frac{|a(\rho^1_\sigma)|n(\overline{\rho^2}_{c_1,g}) + |a(\rho^2_\sigma)|n(\overline{\rho^1}_{c_1,g})}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&\geq \frac{n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2\rfloor} \star \rho^2_{c_2,g}\big) + n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&+ \frac{n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor (a(\rho^1_\sigma)+a(\rho^2_\sigma))/2\rfloor+\lfloor a(\rho^2_\sigma)/2\rfloor}\big) + n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor (a(\rho^1_\sigma)+a(\rho^2_\sigma))/2\rfloor+a(\rho^2_\sigma)+\lfloor a(\rho^1_\sigma)/2\rfloor}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&\geq \frac{n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor (a(\rho^1_\sigma)+a(\rho^2_\sigma))/2\rfloor+\lfloor a(\rho^2_\sigma)/2\rfloor} \star (\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2\rfloor} \star \rho^2_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&+ \frac{n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor (a(\rho^1_\sigma)+a(\rho^2_\sigma))/2\rfloor+a(\rho^2_\sigma)+\lfloor a(\rho^1_\sigma)/2\rfloor} \star (\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&= \frac{n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho^2_{c_2,g}\big) + n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor+a(\rho^2_\sigma)} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&\geq \frac{n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho^2_{c_2,g} \star (\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor+a(\rho^2_\sigma)} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&= \frac{n\big((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho^2_{c_2,g} \star (\overline{\rho^1}_{c_1,g})^{a(\rho^2_\sigma)} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)} \\
&= \frac{n\big((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)},
\end{aligned}
$$

where in the second inequality we used

$$n(\overline{\rho}_{c_1,g}) = n(\rho_{c_1,g} \star (\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)^{-1}) \leq n(\rho_{c_1,g}) + n(\overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta)) \leq 2n(\rho_{c_1,g})$$

and Affine Boundedness, in the second-to-last equality we used

$$\overline{\rho}_{c_1,g} = \overline{\rho^2}_{c_1,g} \star \rho^2_{c_2,g} \star \overline{\rho^1}_{c_1,g} \star (\rho^2_{c_2,g})^{-1}$$

(from Algorithm 6 and the fact that $\overline{\mathsf{R}}$ is a commutative normal subgroup), and in the last equality we used

$$\rho_{c_2,g} = \rho^2_{c_2,g} \star (\overline{\rho}^1_{c_1,g})^{a(\rho^2_\sigma)} \star \rho^1_{c_2,g}$$

(from Algorithm 6).

Bound on $n(\mathsf{Gen}(1^\kappa))$.

We need to show that $\rho \leftarrow \mathsf{Gen}(1^\kappa)$ always satisfies $n(\rho) \leq 1$. The first term of the norm follows easily from the UGS scheme's properties, as each input wire and gate output is sampled as in $\mathsf{UGS}.\mathcal{G}(1^\kappa)$.

The second term is worse, as it depends on free-addition bound α. However, it only adds in at most α extra $\overline{\rho}_{\sigma'}$ rerandomization values (where σ' ranges over input wires and output wires of multiplication gates), and each $n(\overline{\rho}_{\sigma'}) \leq 2n(\overline{\rho}_{\sigma'} \star \Delta) \leq 2$ by subadditivity and the fact that key does not increase the norm (so $n(\Delta) \leq n(\overline{\rho}_{\sigma'} \star \Delta)$). Similarly, using subadditivity and the decomposition respecting the norm, $n(\overline{\rho}_{c_1,g}) \leq 2$. Therefore, $n(\rho_{c_1,g}) = n(\overline{\rho}_{c_1,g} \star \overline{\rho}_{\sigma_1} \star \overline{\rho}_{\sigma_2} \star \Delta) \leq 4\alpha + 3$, which cancels with the $\frac{1}{4\alpha+3}$ making the second term of the norm at most 1.

For the third term, by a similar argument to the second we have $n(\rho_{c_2,g}) \leq 4\alpha + 3$. We also have $n(\overline{\rho}_{\sigma'}) \leq 1$ for the output wire of the multiplication gate. Therefore, by Affine Boundedness and subadditivity,

$$n\big((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho_{c_2,g}\big) \leq 2 \cdot \left|\left\lfloor \frac{a(\rho_\sigma)}{2} \right\rfloor\right| + 4\alpha + 3 \leq |a(\rho_\sigma)| + 4\alpha + 4 \leq \beta_{\mathsf{MAC}} + 4\alpha + 4,$$

so

$$\sqrt{\frac{n\big((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}} \leq \sqrt{\frac{\beta_{\mathsf{MAC}}+4\alpha+4}{2\beta_{\mathsf{MAC}}(4\alpha+3)}} \leq 1,$$

since $\beta_{\mathsf{MAC}} \geq 1$ and $\alpha \geq 0$.

We also need to show similar bounds for $n(\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C}, \beta))$. We decompose the output of $\mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C}, \beta)$ as $\rho = \rho^2 \star \rho^1$, where ρ^2 contains only local ciphertext rerandomization terms $\overline{\rho}_{c_1,g}$ and $\overline{\rho}_{c_2,g}$, and ρ^1 only contains wire and global rerandomization terms. We then use subadditivity $n(\rho) \leq n(\rho^2) + n(\rho^1)$, and bound the two G-seminorms separately.

First, we bound $n(\rho^1)$. For the first term of the RGS norm, we get

$$\begin{aligned}
\rho^1_\sigma &= \overline{\rho^1}_\sigma \star \Delta \\
n(\rho^1_\sigma) &\leq n(\overline{\rho^1}_\sigma) + n(\Delta) \\
&\leq \beta_{\mathrm{mask}} 2(\beta_{\mathrm{mask}}+1)\beta + \beta_{\mathrm{mask}}\beta \\
&= (2\beta_{\mathrm{mask}}+3)\beta_{\mathrm{mask}}\beta.
\end{aligned}$$

The other terms are no bigger, as $n(\rho^1_{c_1}) = n(\rho^1_{c_2}) = n(\rho^1_\sigma)$, so $n(\rho^1) \leq (2\beta_{\mathrm{mask}}+3)\beta_{\mathrm{mask}}\beta$.

Next, we bound $n(\rho^2)$. The first term of the RGS norm is 0, as ρ^2 does not rerandomize the wires. For the second term of the RGS norm, we get

$$\begin{aligned}
\rho^2_{c_1,g} &= \overline{\rho^2}_{c_1,g} \star \overline{\rho^2}_{\sigma_1} \star \overline{\rho^2}_{\sigma_2} \star \Delta \\
n(\rho^2_{c_1,g}) &\leq n(\overline{\rho^2}_{c_1,g}) + n(\overline{\rho^2}_{\sigma_1}) + n(\overline{\rho^2}_{\sigma_2}) + n(\Delta) \\
&\leq \beta_{\mathrm{mask}}(8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}} + 8\alpha + 6)\beta + 2\beta_{\mathrm{mask}}2(\beta_{\mathrm{mask}}+1)\beta + \beta_{\mathrm{mask}}\beta \\
&= (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}} + 8\alpha + 6 + 4(\beta_{\mathrm{mask}}+1) + 1)\beta_{\mathrm{mask}}\beta \\
&= (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+6)\beta_{\mathrm{mask}} + 8\alpha + 11)\beta_{\mathrm{mask}}\beta \\
\frac{n(\rho^2_\sigma)}{4\alpha+3} &\leq \left(\frac{8\alpha}{4\alpha+3}\beta^2_{\mathrm{mask}} + 2\beta_{\mathrm{mask}} + \frac{8\alpha+11}{4\alpha+3}\right)\beta_{\mathrm{mask}}\beta \\
&\leq 2(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta_{\mathrm{mask}}\beta.
\end{aligned}$$

For the third term of the RGS norm, we get

$$\begin{aligned}
&\rho^2_{c_2,g} = \overline{\rho^2}_{c_2,g} \star \overline{\rho^2}_{\sigma_1} \star \overline{\rho^2}_{\sigma_2} \star \Delta \\
&n\big((\overline{\rho^2}_{c_1,g})^{-\lfloor a(\rho^2_\sigma)/2\rfloor} \star \rho^2_{c_2,g}\big) \\
&\quad = n(\rho^2_{c_1,g}) \\
&\quad \leq n(\overline{\rho^2}_{c_1,g}) + n(\overline{\rho^2}_{\sigma_1}) + n(\overline{\rho^2}_{\sigma_2}) + n(\Delta) \\
&\quad \leq \beta_{\mathrm{mask}}((8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}})\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta^2) \\
&\quad + 2\beta_{\mathrm{mask}}2(\beta_{\mathrm{mask}}+1)\beta + \beta_{\mathrm{mask}}\beta \\
&\quad = (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}})\beta_{\mathrm{mask}}\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta_{\mathrm{mask}}\beta^2 \\
&\quad + (4(\beta_{\mathrm{mask}}+1)+1)\beta_{\mathrm{mask}}\beta \\
&\quad = (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+6)\beta_{\mathrm{mask}} + 4)\beta_{\mathrm{mask}}\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta_{\mathrm{mask}}\beta^2 \\
&\sqrt{\frac{n\big((\overline{\rho}_{c_1,g})^{-\lfloor a(\rho_\sigma)/2\rfloor} \star \rho_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}} \\
&\quad \leq \sqrt{\frac{(8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+6)\beta_{\mathrm{mask}} + 4)\beta_{\mathrm{mask}}\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta_{\mathrm{mask}}\beta^2}{2\beta_{\mathsf{MAC}}(4\alpha+3)}} \\
&\quad = \sqrt{\left(\frac{4\alpha\beta^2_{\mathrm{mask}}}{\beta_{\mathsf{MAC}}(4\alpha+3)} + \frac{\beta_{\mathrm{mask}}}{\beta_{\mathsf{MAC}}} + \frac{2}{\beta_{\mathsf{MAC}}(4\alpha+3)}\right)\beta_{\mathrm{mask}}\beta + 2(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta_{\mathrm{mask}}\beta^2}
\end{aligned}$$

$$\leq \sqrt{(\beta_{\text{mask}}^2 + \beta_{\text{mask}} + 1)\beta_{\text{mask}}\beta + 2(\beta_{\text{mask}}^2 + \beta_{\text{mask}} + 2)\beta_{\text{mask}}\beta^2}$$
$$\leq \sqrt{3(\beta_{\text{mask}}^2 + \beta_{\text{mask}} + 2)\beta_{\text{mask}}\beta}$$
$$\leq 2(\beta_{\text{mask}}^2 + \beta_{\text{mask}} + 2)\beta_{\text{mask}}\beta,$$

since we can assume $\beta \geq 1$. Therefore,

$$\begin{aligned} n(\rho) &\leq n(\rho^2) + n(\rho^1) \\ &\leq 2(\beta_{\text{mask}}^2 + \beta_{\text{mask}} + 2)\beta_{\text{mask}}\beta + (2\beta_{\text{mask}} + 3)\beta_{\text{mask}}\beta \\ &= (2\beta_{\text{mask}}^2 + 4\beta_{\text{mask}} + 5)\beta_{\text{mask}}\beta, \end{aligned}$$

so the RGS scheme works with $\beta'_{\text{mask}} = (2\beta_{\text{mask}}^2 + 4\beta_{\text{mask}} + 5)\beta_{\text{mask}}$.

Masking.

Next, we must show that for all ρ^1 in R_{re} with $n(\rho^1) \leq \beta$,

$$\{\rho^2 \star \rho^1\}_{\rho^2 \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)} \overset{\text{s}}{\approx} \{\rho^2\}_{\rho^2 \leftarrow \mathsf{Gen}_{\mathsf{mask}}(1^\kappa, \beta)},$$

according to the algorithms $\mathsf{Gen}_{\mathsf{mask}}$ and $\star$ defined in Algorithm 6. This proof follows a similar approach to Appendix D.2 – we present a statistical hybrid argument from the real distribution to the random distribution, and justify each step using Lemma 22.

Hybrid_1: This is the real distribution. Let $\rho' = \rho^2 \star \rho^1$ be the output of the group operation. For $i \in \{1, 2, \prime\}$ parse ρ^i as the tuple $(\Delta^i, \{\overline{\rho}^i_\sigma, \overline{\rho}^i_{c_1,g}, \overline{\rho}^i_{c_2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}^i_\sigma\}_{\sigma \in [\mathsf{n}]})$. For all $i \in \{1, 2, \prime\}$ and all wires σ, let $\rho^i_\sigma = \overline{\rho}^i_\sigma \star \Delta^i$, and for all gates $g \in [\mathsf{Q}]$ with input and output wires $(\sigma_1, \sigma_2, \sigma)$ and all $j \in \{1, 2\}$, let $\rho^i_{c_j,g} = \overline{\rho}^i_{c_j,g} \star \overline{\rho}^i_{\sigma_1} \star \overline{\rho}^i_{\sigma_2} \star \Delta^i$. According to Algorithm 6, the output is generated as

$$\begin{aligned} \Delta' &= \Delta^2 \star \Delta^1 \\ \forall \sigma \in [\mathsf{n}], \quad & \rho'_\sigma = \rho^2_\sigma \star \rho^1_\sigma \\ \forall g \in [\mathsf{Q}], \text{ if } \mathsf{op}_g = +, \quad & \overline{\rho}'_\sigma = \overline{\rho}'_{\sigma_1} \star \overline{\rho}'_{\sigma_2} \\ \text{if } \mathsf{op}_g = \times, \quad & \rho'_\sigma = \rho^2_\sigma \star \rho^1_\sigma \\ & \rho'_{c_1,g} = \rho^2_{c_1,g} \star \rho^1_{c_1,g} \\ & \rho'_{c_2,g} = \rho^2_{c_2,g} \star (\overline{\rho}^1_{c_1,g})^{a(\rho^2_\sigma)} \star \rho^1_{c_2,g}, \end{aligned}$$

where $a(\rho^2_\sigma)$ is defined as in the proof for the Group Action above, and the group elements have decompositions $\rho^1_{c_2,g} = \overline{\rho}^1_{c_2,g} \star \overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1$, etc.

Hybrid_2: In this hybrid, we remove all the $(\overline{\rho}^1_{c_1,g})^{a(\rho^2_\sigma)}$ and $\overline{\rho}^1_{c_2,g}$ terms, and instead set $\rho'_{c_2,g} = \rho^2_{c_2,g} \star \overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1$. That is, we use Lemma 22 once for each multiplication gate with $\rho^*_c = \overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2$ and $\overline{\rho}_c = (\overline{\rho}^1_{c_1,g})^{a(\rho^2_\sigma)} \star \overline{\rho}^1_{c_2,g}$.

To calculate the norms, we will need upper bounds on $n(\rho^1_\sigma)$, $n(\rho^1_{c_1,g})$, and $n(\rho^1_{c_2,g})$ in terms of β. The first two terms of the definition of n for the RGS give

$$\begin{aligned} n(\rho^1_\sigma) &\leq \beta \qquad \text{for input wires and outputs of multiplication gates} \\ n(\rho^1_{c_1,g}) &\leq (4\alpha + 3)\beta \end{aligned}$$

For $\rho^1_{c_2,g}$, we get

$$\beta^2 \geq \max\left(n(\rho^1_\sigma)^2, \left(\frac{n(\rho^1_{c_1,g})}{4\alpha + 3}\right)^2, \frac{n\left((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2 \rfloor} \star \rho^1_{c_2,g}\right)}{2\beta_{\mathsf{MAC}}(4\alpha + 3)} \right)$$

$$
\begin{aligned}
&\geq \max\left(\frac{n(\rho^1_\sigma)n(\rho^1_{c_1,g})}{4\alpha+3}, \frac{n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}\right)\\
&\geq \max\left(\frac{|a(\rho^1_\sigma)|n(\overline{\rho^1}_{c_1,g})}{2\beta_{\mathsf{MAC}}(4\alpha+3)}, \frac{n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}\right)\\
&\geq \max\left(\frac{|\lfloor a(\rho^1_\sigma)/2\rfloor|n(\overline{\rho^1}_{c_1,g})}{2\beta_{\mathsf{MAC}}(4\alpha+3)}, \frac{n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{2\beta_{\mathsf{MAC}}(4\alpha+3)}\right)\\
&\geq \frac{|\lfloor a(\rho^1_\sigma)/2\rfloor|n(\overline{\rho^1}_{c_1,g}) + n\big((\overline{\rho^1}_{c_1,g})^{-\lfloor a(\rho^1_\sigma)/2\rfloor} \star \rho^1_{c_2,g}\big)}{4\beta_{\mathsf{MAC}}(4\alpha+3)}\\
&\geq \frac{n\big(\rho^1_{c_2,g}\big)}{4\beta_{\mathsf{MAC}}(4\alpha+3)}\\
n\big(\rho^1_{c_2,g}\big) &\leq 4\beta_{\mathsf{MAC}}(4\alpha+3)\beta^2
\end{aligned}
$$

We can now show that the norm condition is satisfied:

$$
\begin{aligned}
&2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + n((\overline{\rho}^1_{c_1,g})^{a(\rho^2_\sigma)} \star \overline{\rho}^1_{c_2,g})\\
&\quad= 2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + n((\rho^1_{c_1,g} \star (\Delta^1)^{-1})^{a(\rho^2_\sigma)} \star \rho^1_{c_2,g} \star (\overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1)^{-1})\\
&\quad\leq 2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + 2|a(\rho^2_\sigma)| \cdot n(\rho^1_{c_1,g}) + 2n(\rho^1_{c_2,g})\\
&\quad\leq 2(2(\alpha\beta_{\mathrm{mask}} \cdot 2(\beta_{\mathrm{mask}}+1)\beta) + \beta_{\mathrm{mask}}\beta)\\
&\quad+ 2\beta_{\mathsf{MAC}}\beta_{\mathrm{mask}}2(\beta_{\mathrm{mask}}+1)\beta \cdot (4\alpha+3)\beta + 2 \cdot 4\beta_{\mathsf{MAC}}(4\alpha+3)\beta^2\\
&\quad= (8\alpha\beta^2_{\mathrm{mask}} + 8\alpha\beta_{\mathrm{mask}} + 2\beta_{\mathrm{mask}})\beta\\
&\quad+ 4\beta_{\mathsf{MAC}}\beta_{\mathrm{mask}}(\beta_{\mathrm{mask}}+1)(4\alpha+3)\beta^2 + 8\beta_{\mathsf{MAC}}(4\alpha+3)\beta^2\\
&\quad= (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}})\beta + 4(4\alpha+3)\beta_{\mathsf{MAC}}(\beta^2_{\mathrm{mask}} + \beta_{\mathrm{mask}} + 2)\beta^2.
\end{aligned}
$$

Here, we have used that at most α (with duplication) masks sampled from $\mathsf{UGS}.\mathcal{G}^*_{\mathsf{MAC}}$ get added together in each of $\overline{\rho}^2_{\sigma_1}$ and $\overline{\rho}^2_{\sigma_2}$, and used Affine Boundedness to upper bound $a(\rho^2_\sigma)$. Therefore, this change is statistically indistinguishable.

Hybrid_3: In this hybrid, we remove all $\overline{\rho}^1_{c_j,g}$ terms, and instead let $\rho'_{c_1,g} = \rho^2_{c_1,g} \star \overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1$. This is indistinguishable by Lemma 22 used once for each multiplication gate, with $\rho^*_c = \overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2$ and $\overline{\rho}_c = \overline{\rho}^1_{c_1,g}$. The norm condition is satisfied:

$$
\begin{aligned}
&2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + n(\overline{\rho}^1_{c_1,g})\\
&\quad= 2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + n(\rho^1_{c_1,g} \star (\overline{\rho}^1_{\sigma_1} \star \overline{\rho}^1_{\sigma_2} \star \Delta^1)^{-1})\\
&\quad\leq 2n(\overline{\rho}^2_{\sigma_1} \star \overline{\rho}^2_{\sigma_2} \star \Delta^2) + 2n(\rho^1_{c_1,g})\\
&\quad\leq 2(2(\alpha\beta_{\mathrm{mask}} \cdot 2(\beta_{\mathrm{mask}}+1)\beta) + \beta_{\mathrm{mask}}\beta) + 2(4\alpha+3)\beta\\
&\quad= (8\alpha\beta_{\mathrm{mask}}(\beta_{\mathrm{mask}}+1) + 2\beta_{\mathrm{mask}} + 8\alpha + 6)\beta\\
&\quad= (8\alpha\beta^2_{\mathrm{mask}} + (8\alpha+2)\beta_{\mathrm{mask}} + 8\alpha + 6)\beta.
\end{aligned}
$$

Here, we have used that at most α (with duplication) masks sampled from $\mathsf{UGS}.\mathcal{G}^*_{\mathsf{MAC}}$ get added together in each of $\overline{\rho}^2_{\sigma_1}$ and $\overline{\rho}^2_{\sigma_2}$. Therefore, this change is statistically indistinguishable.

Hybrid_4: Use $\overline{\rho}'_{\sigma_1}$ and $\overline{\rho}'_{\sigma_2}$ in the computation of $\rho'_{c_1,g}$ and $\rho'_{c_2,g}$, instead of $\overline{\rho}^1_{\sigma_1}$ and $\overline{\rho}^2_{\sigma_1}$, etc. This is identical because $\overline{\mathsf{R}}_{\mathsf{MAC}}$ is commutative.

Hybrid_5: Next, remove all $\overline{\rho}^1_\sigma$ terms, and instead let $\rho'_\sigma = \rho^2_\sigma \star \Delta^1$. Note that since the previous hybrid, $\overline{\rho}^1_\sigma$ is only used to compute ρ'_σ. This is indistinguishable by Lemma 22 used once for

each input wire and once for each multiplication gate, with $\rho^*_\sigma = \Delta^2$ and $\overline{\rho}_\sigma = \overline{\rho}^1_\sigma$. The norm condition is satisfied:

$$\begin{aligned} 2n(\Delta^2) + n(\overline{\rho}^1_\sigma) &= 2n(\Delta^2) + n(\rho^1_\sigma \star (\Delta^1)^{-1}) \\ &\leq 2\beta_{\text{mask}}\beta + 2\beta \\ &= 2(\beta_{\text{mask}} + 1)\beta. \end{aligned}$$

Therefore, this change is statistically indistinguishable.

Hybrid_6: Since the last hybrid, everywhere Δ^1 is used is of the form $\Delta^2 \star \Delta^1$. Replace this expression with just Δ^2. This is indistinguishable by masking, since $n(\Delta^1) = n(\mathsf{key}(\rho^1_\sigma)) \leq \beta$. We are now at the random distribution.

Rerand-Correctness.

The Rerand-correctness property of RGS (Definition 3) holds due to (1) the correctness property of UGS (Definition 8) holding for all choices of randomness in UGS.R; and (2) the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ of functions in the function family $\mathcal{F}_{\mathsf{pp}}$ of the UGS being set such that the values $\overline{\boldsymbol{x}}_\sigma$ (offset input) and X_σ (MAC) for any circuit wire σ never exceeds $|\mathcal{X}|$. The latter holds as long as the wire value (during evaluation) does not exceed the bound B and the values of the mask $\boldsymbol{m}_\sigma$ and global public key pk are sampled using randomness ρ from R_{re} where $n(\rho) \leq T$. This ensures that within the function family $\mathcal{F}_{\mathsf{pp}}$ of the UGS, where the input space $\mathcal{X}$ is set as a function of B and the size of the space R, evaluating the bilinear function $f(x)$ will not wrap around the modulus. Note that this holds even when the inputs to f are outputs of addition gates due to the linear property of the function UGS.MAC and the fact that adding offset inputs is also linear. Therefore the inputs to f are always from the correct space, $f(x) = f(x) \mod q$ and this ensures correctness of the overall B-bounded arithmetic computation evaluated by the garbling scheme.

Rerand-Privacy.

We show that the garbling scheme RGS (Algorithms 4 to 6) satisfies Rerand-privacy (Definition 3) via reduction to the security properties of UGS given in Definition 10.

Simulation. Let $\mathcal{S}_{\mathsf{Enc}}$, $\mathcal{S}_{\mathsf{MAC}}$ and $\mathsf{Sim}_{\mathsf{UGS}}$ be the simulators for 'ciphertext simulation', 'MAC simulation' and 'privacy' from Definition 10, respectively. The PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ for RGS uses the algorithms $\mathcal{S}_{\mathsf{MAC}}$ and $\mathsf{Sim}_{\mathsf{UGS}}$ in a black-box way and acts as given in Algorithm 9.

Hybrids. Let the circuit $\mathbf{C}$ being computed have $\mathsf{Q}_\times$ multiplication gates. In order to show that the distribution of the output of the simulation in Algorithm 9 is computationally indistinguishable from the real execution in the Rerand-privacy security game (Definition 3), we consider the following set of intermediate hybrid experiments:

1. Hybrid_0 : This is the hybrid experiment whose output is distributed identically as in the simulation Sim described in Algorithm 9.

2. Hybrid_1 : This is an intermediate hybrid experiment that is described in Algorithm 10. This experiment differs from the simulation only in that the function input $\boldsymbol{x}$ is used to compute the real values $\boldsymbol{x}_\sigma$ within each wire σ of the circuit $\mathbf{C}$, instead of assigning all of them the value 0 as in Sim. The rest of the experiment is executed in the same way. In our proof we will show that the output of this experiment is distributed statistically closely to that of the simulation due to the 'input masking' property (Definition 8) of the underlying UGS.

Algorithm 9 Simulator for RGS

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, where $\mathcal{Y}$ is a module over a ring $\mathbb{R}$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8 and 10). Then $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ is defined as follows:

1. $\mathsf{Sim}_0(\rho_{\mathbf{C}}) \to (\boldsymbol{F}, \boldsymbol{X}, \mathsf{state})$
   - let $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g\in[\mathsf{Q}]})$ be a circuit representing f
   - let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be default public values for UGS
   - parse $\rho_{\mathbf{C}}$ as $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - sample $\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)$
   - set $\mathsf{pk} = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)$
   - set $\boldsymbol{x} = 0^{\mathsf{n}}$ and for all $\sigma \in [\mathsf{n}]$,
     - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
     - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
     - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
   - for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
     if $\mathsf{op}_g == +$,
     - set $\mathsf{pk}_\sigma = \mathsf{pk}_{\sigma_1} + \mathsf{pk}_{\sigma_2}$, $\boldsymbol{m}'_\sigma = \boldsymbol{m}'_{\sigma_1} + \boldsymbol{m}'_{\sigma_2}$, $X_\sigma = X_{\sigma_1} + X_{\sigma_2}$, and $\overline{\boldsymbol{x}}_\sigma = \overline{\boldsymbol{x}}_{\sigma_1} + \overline{\boldsymbol{x}}_{\sigma_2}$

     if $\mathsf{op}_g == \times$,
     - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
     - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$ and $\boldsymbol{x}_\sigma = 0$
     - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
     - compute $c'_{1,g} \leftarrow \mathsf{Sim}_{\mathsf{UGS}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, \overline{\boldsymbol{x}}, \boldsymbol{X}, \overline{\boldsymbol{x}}_\sigma)$
     - compute $c'_{2,g} \leftarrow \mathsf{Sim}_{\mathsf{UGS}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, \overline{\boldsymbol{x}}, \boldsymbol{X}, X_\sigma)$
   - set $\boldsymbol{F} = (\mathsf{pk}, \{(c'_{1,g}, c'_{2,g}, \mathsf{pk}_\sigma)\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\mathsf{pk}_\sigma\}_{\sigma\in[\mathsf{n}]})$
   - set $\boldsymbol{X} = \{(\overline{\boldsymbol{x}}_\sigma, X_\sigma)\}_{\sigma\in[\mathsf{n}]}$
   - set $\mathsf{state} = \{\boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$
   - return $(\boldsymbol{F}, \boldsymbol{X}, \mathsf{state})$
2. $\mathsf{Sim}_1(\mathbf{C}(\boldsymbol{x}), \mathsf{state}) \to \boldsymbol{d}$
   - parse $\mathsf{state} = \{\boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$
   - return $\boldsymbol{d} = \{\boldsymbol{m}_\sigma - \mathbf{C}(\boldsymbol{x})_\sigma\}_{\sigma\in[\mathsf{m}]}$

3. Hybrid_2 : This is an intermediate hybrid experiment that is described in Algorithm 11. This experiment differs from Hybrid_1 in that the garbling for each multiplication gate is generated by using $\mathcal{S}_{\mathsf{Enc}}$ to simulate the two ciphertexts. In our proof we will show that the output of this experiment is computationally indistinguishable from that of Hybrid_1 via $\mathsf{Q}_\times$ intermediate hybrid experiments where each pair of adjacent hybrids can be shown computationally indistinguishable due to the 'privacy' property (Definition 10) of the underlying UGS.

4. Hybrid_3 : This is an intermediate hybrid experiment that is described in Algorithm 12. This experiment differs from Hybrid_2 in that all circuit input wire labels, as well as output wire labels $(\overline{\boldsymbol{x}}_\sigma, X_\sigma)$ of multiplication gates are generated as in the real execution (using $\mathsf{UGS.MAC}$), as opposed to using $\mathcal{S}_{\mathsf{MAC}}$ as in the simulation. In our proof we will show that the output of this experiment is statistically indistinguishable from that of Hybrid_2

via $\mathsf{Q}_\times + \mathsf{n}$ intermediate hybrid experiments where each pair of adjacent hybrids can be shown as statistically close due to the 'MAC simulation' property (Definition 10) of the underlying UGS.

5. Hybrid_4 : The output of this experiment is identically distributed to the real execution in the Rerand-privacy security game (Definition 3). This hybrid experiment differs from Hybrid_3 only in that the ciphertexts for all the multiplication gates are generated as in RGS, instead of using $\mathcal{S}_{\mathsf{Enc}}$. In our proof we will show that this distribution in computationally indistinguishable from from the output of Hybrid_3 via $2\mathsf{Q}_\times$ intermediate hybrid experiments where each pair of adjacent hybrids can be shown as computationally indistinguishable by reduction to the 'ciphertext simulation' property (Definition 10) of the underlying UGS.

Algorithm 10 Experiment Hybrid_1 for RGS

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, where $\mathcal{Y}$ is a module over a ring $\mathbb{R}$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8 and 10). For a circuit $\mathbf{C}$ with input $\boldsymbol{x}^{\mathsf{n}} \in \mathcal{X}^{\mathsf{n}}$, the experiment Hybrid_1 operates as follows:

$\mathsf{Hybrid}_1(\rho_{\mathbf{C}}, \mathbf{C}, \boldsymbol{x}^{\mathsf{n}}) \to (\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

- let $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g\in[\mathsf{Q}]})$ be the circuit
- for each wire $\sigma \in [\mathsf{n}+\mathsf{Q}]$, use $\mathbf{C}, \boldsymbol{x}^{\mathsf{n}}$ to compute the real wire values $\boldsymbol{x}_\sigma$
- let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be default public values for UGS
- parse $\rho_{\mathbf{C}}$ as $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\overline{\rho}_\sigma\}_{\sigma\in[\mathsf{n}]})$
- sample $\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)$
- set $\mathsf{pk} = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)$
- for all $\sigma \in [\mathsf{n}]$,
  - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
- for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n}+\mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
  if $\mathsf{op}_g == +$,
  - set $\mathsf{pk}_\sigma = \mathsf{pk}_{\sigma_1} + \mathsf{pk}_{\sigma_2}$, $\boldsymbol{m}'_\sigma = \boldsymbol{m}'_{\sigma_1} + \boldsymbol{m}'_{\sigma_2}$, $X_\sigma = X_{\sigma_1} + X_{\sigma_2}$, and $\overline{\boldsymbol{x}}_\sigma = \overline{\boldsymbol{x}}_{\sigma_1} + \overline{\boldsymbol{x}}_{\sigma_2}$

  if $\mathsf{op}_g == \times$,
  - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
  - compute $c'_{1,g} \leftarrow \mathsf{Sim}_{\mathsf{UGS}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, \overline{\boldsymbol{x}}, \boldsymbol{X}, \overline{\boldsymbol{x}}_\sigma)$
  - compute $c'_{2,g} \leftarrow \mathsf{Sim}_{\mathsf{UGS}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, \overline{\boldsymbol{x}}, \boldsymbol{X}, X_\sigma)$
- set $\boldsymbol{F} = (\mathsf{pk}, \{(c'_{1,g}, c'_{2,g}, \mathsf{pk}_\sigma)\}_{g\in[\mathsf{Q}] \text{ s.t. } \mathsf{op}_g=\times}, \{\mathsf{pk}_\sigma\}_{\sigma\in[\mathsf{n}]})$
- set $\boldsymbol{X} = \{(\overline{\boldsymbol{x}}_\sigma, X_\sigma)\}_{\sigma\in[\mathsf{n}]}$
- set $\boldsymbol{d} = \{\boldsymbol{m}_\sigma\}_{\sigma\in[\mathsf{m}]}$
- return $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

Claim 23. *Assuming* UGS *satisfies 'input masking' (Definition 8),* $\mathsf{Hybrid}_0 \overset{\mathrm{c}}{\approx} \mathsf{Hybrid}_1$

Proof. The hybrid experiments Hybrid_0 and Hybrid_1 differ only in how the wire values $\boldsymbol{x}_\sigma$ are assigned. In Hybrid_0, all the circuit wires are assigned the value $\boldsymbol{x}_\sigma = 0$, while in Hybrid_1 the

circuit $\mathbf{C}$ and input $\boldsymbol{x}$ are used to compute the real values $\boldsymbol{x}_\sigma$ that each wire σ takes within the garbling evaluation. Consider for the sake of contradiction that the output distributions of these hybrids were distinguishable with non-negligible advantage ϵ. Then there must exist at least one circuit wire $\sigma \in [\mathsf{n} + \mathsf{Q}]$ for which the distributions, $\{m_\sigma + \boldsymbol{x}_\sigma\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}}$ (as in Hybrid_1) can be distinguished from $\{m_\sigma\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}}$ (as in Hybrid_0) with advantage $\geq \frac{\epsilon}{\mathsf{n}+\mathsf{Q}}$, which is also non-negligible. However, as long as $x \in \mathcal{X}'$ (as will always be the case for B-bounded arithmetic computation), it holds due to 'input masking' (Definition 8) that,

$$\{m + x\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa)} \overset{\mathrm{s}}{\approx} \{m\}_{m \leftarrow \mathcal{G}_{\mathsf{offset}}(1^\kappa)}$$

thereby yielding a contradiction. In particular, we have $n(\rho_\sigma) \leq n(\rho) \leq T$, so it satisfies the condition for using MAC simulation. □

Claim 24. *Assuming* UGS *satisfies 'privacy' (Definition 10),* $\mathsf{Hybrid}_1 \overset{\mathrm{c}}{\approx} \mathsf{Hybrid}_2$

Proof. The hybrid experiments Hybrid_1 and Hybrid_2 differ in how the two ciphertexts for each multiplication gate have been created. In Hybrid_1, the ciphertexts are generated using the simulator $\mathsf{Sim}_{\mathsf{UGS}}$ that does not receive the function $f \in \mathcal{F}^*_{\mathsf{pp}}$ and input $\boldsymbol{x}$, while in Hybrid_2, these ciphertexts are generated using the algorithm $\mathcal{S}_{\mathsf{Enc}}$ that receives these values in addition to the inputs that $\mathsf{Sim}_{\mathsf{UGS}}$ receives. In order to show that these distributions are indistinguishable, let us consider the following set of hybrid experiments:

- Hybrid'_0 : this is distributed exactly like the experiment in Hybrid_1.
- $\forall i \in [\mathsf{Q}_\times]$, Hybrid'_{2i-1} : this differs from Hybrid_1 in that all the ciphertexts for up to the $i-1^{th}$ multiplication gate is generated using $\mathcal{S}_{\mathsf{Enc}}$. Among the ciphertexts (c_1, c_2) of the ith multiplication gate, c_1 is generated using $\mathcal{S}_{\mathsf{Enc}}$, while c_2 is generated using $\mathsf{Sim}_{\mathsf{UGS}}$.
- $\forall i \in [\mathsf{Q}_\times]$, Hybrid'_{2i} : this differs from Hybrid_1 in that all the ciphertexts for up to the ith multiplication gate is generated using $\mathcal{S}_{\mathsf{Enc}}$.
- Note that the last of the above hybrid experiments is distributed exactly like in Hybrid_2.

In the above, each set of adjacent hybrids differ only in how one of the ciphertexts is created. Consider for the sake of contradiction that there exists a PPT distinguisher D that can distinguish between Hybrid'_0 and $\mathsf{Hybrid}'_{2\mathsf{Q}_\times}$ with non-negligible advantage ϵ. Then there must exist an $j \in [2\mathsf{Q}_\times]$ such that D can distinguish between Hybrid'_{j-1} and Hybrid'_j with probability $> \frac{\epsilon}{2\mathsf{Q}_\times}$. We show that D can be used in a black-box way by a PPT adversary $\mathcal{A}_{\mathsf{UGS}}$ to distinguish in the 'privacy' game (Definition 10), as follows:

- $\mathcal{A}_{\mathsf{UGS}}$ samples garbling randomnesses $\rho_{\mathbf{C}}$ and $\rho'_{\mathbf{C}}$ and computes the garbling following the hybrid experiment Hybrid_2, until the $i-1^{th}$ multiplication gate is created.
- Set $f_1(\Delta)$ and $f_2(\Delta)$ as described in Algorithm 11.
- For the ith multiplication gate, if the ciphertext at which the hybrids differ is the first ciphertext, $\mathcal{A}_{\mathsf{UGS}}$ gives to the challenger $\mathsf{C}_{\mathsf{UGS}}$ the tuple $(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, f_1(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f_1)$ and receives a ciphertext c_b. $\mathcal{A}_{\mathsf{UGS}}$ uses c_b as this first ciphertext in the multiplication gate and generates the second ciphertext using $\mathsf{Sim}_{\mathsf{UGS}}$.
- Otherwise, $\mathcal{A}_{\mathsf{UGS}}$ creates the first ciphertext using $\mathcal{S}_{\mathsf{Enc}}$ with $(f_1(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f_1)$ and then gives to the challenger $\mathsf{C}_{\mathsf{UGS}}$ the tuple $(\mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, f_2(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f_2)$ and receives a ciphertext c_b. $\mathcal{A}_{\mathsf{UGS}}$ uses c_b as the second ciphertext in the multiplication gate.
- $\mathcal{A}_{\mathsf{UGS}}$ completes creating $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ by computing the rest of the ciphertexts following the hybrid experiment Hybrid_{2i-1}.

Algorithm 11 Experiment Hybrid_2 for RGS

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, where $\mathcal{Y}$ is a module over a ring $\mathbb{R}$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8 and 10). For a circuit $\mathbf{C}$ with input $\boldsymbol{x}^{\mathsf{n}} \in \mathcal{X}^{\mathsf{n}}$, the experiment Hybrid_2 operates as follows:

$\mathsf{Hybrid}_2(\rho_{\mathbf{C}}, \mathbf{C}, \boldsymbol{x}^{\mathsf{n}}) \to (\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

- let $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$ be the circuit
- for each wire $\sigma \in [\mathsf{n} + \mathsf{Q}]$, use $\mathbf{C}, \boldsymbol{x}^{\mathsf{n}}$ to compute the real wire values $\boldsymbol{x}_\sigma$
- let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be default public values for UGS
- parse $\rho_{\mathbf{C}}$ as $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}_\sigma\}_{\sigma \in [\mathsf{n}]})$
- sample $\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)$
- set $\mathsf{pk} = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk}}(\Delta' \star \Delta, \mathsf{pk}^*)$
- for all $\sigma \in [\mathsf{n}]$,
  - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
- for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
  if $\mathsf{op}_g == +$,
  - set $\mathsf{pk}_\sigma = \mathsf{pk}_{\sigma_1} + \mathsf{pk}_{\sigma_2}$, $\boldsymbol{m}'_\sigma = \boldsymbol{m}'_{\sigma_1} + \boldsymbol{m}'_{\sigma_2}$, $X_\sigma = X_{\sigma_1} + X_{\sigma_2}$, and $\overline{\boldsymbol{x}}_\sigma = \overline{\boldsymbol{x}}_{\sigma_1} + \overline{\boldsymbol{x}}_{\sigma_2}$

  if $\mathsf{op}_g == \times$,
  - $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \boldsymbol{m}'_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $(\mathsf{pk}_\sigma, X_\sigma) \leftarrow \mathcal{S}_{\mathsf{MAC}}(\rho_\sigma, \mathsf{pk}, \overline{\boldsymbol{x}}_\sigma)$
  - let $f_1(\widehat{\Delta}) \in \mathcal{F}^*_{\mathsf{pp}}$ be a function s.t.
    $f_1(\widehat{\Delta})(x_1, x_2) = (x_1 - \mathsf{offset}(\rho_{\sigma_1})) \times (x_2 - \mathsf{offset}(\rho_{\sigma_2})) + \mathsf{offset}(\rho_\sigma)$
  - let $f_2(\widehat{\Delta}) \in \mathcal{F}^*_{\mathsf{pp}}$ be a key-dependent function s.t.
    $f_2(\widehat{\Delta})(x_1, x_2) = \mathsf{UGS}.\mathsf{MAC}(\overline{\rho}_\sigma \star \widehat{\Delta}, f_1(\widehat{\Delta})(x_1, x_2))$
  - compute $c'_{1,g} \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, (\overline{\boldsymbol{x}}_{\sigma_1}, \overline{\boldsymbol{x}}_{\sigma_2}), (\boldsymbol{X}_{\sigma_1}, \boldsymbol{X}_{\sigma_2}), \overline{\boldsymbol{x}}_\sigma, (\boldsymbol{x}_{\sigma_1}, \boldsymbol{x}_{\sigma_2}), f_1)$
  - compute $c'_{2,g} \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, (\overline{\boldsymbol{x}}_{\sigma_1}, \overline{\boldsymbol{x}}_{\sigma_2}), (\boldsymbol{X}_{\sigma_1}, \boldsymbol{X}_{\sigma_2}), X_\sigma, (\boldsymbol{x}_{\sigma_1}, \boldsymbol{x}_{\sigma_2}), f_2)$
- set $\boldsymbol{F} = (\mathsf{pk}, \{(c'_{1,g}, c'_{2,g}, \mathsf{pk}_\sigma)\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\mathsf{pk}_\sigma\}_{\sigma \in [\mathsf{n}]})$
- set $\boldsymbol{X} = \{(\overline{\boldsymbol{x}}_\sigma, X_\sigma)\}_{\sigma \in [\mathsf{n}]}$
- set $\boldsymbol{d} = \{\boldsymbol{m}_\sigma\}_{\sigma \in [\mathsf{m}]}$
- return $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

- Note that, if $b = 0$, $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ is distributed as in Hybrid'_{j-1} and otherwise, $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ is distributed as in Hybrid'_j.
- $\mathcal{A}_{\mathsf{UGS}}$ sends $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ to D and outputs whatever it outputs.

In this execution, $\mathcal{A}_{\mathsf{UGS}}$ has the same advantage as D, that is non-negligible. However, since UGS satisfies 'privacy' (Definition 10), no such $\mathcal{A}_{\mathsf{UGS}}$ can exist, and therefore no such D can exist. This yields a contradiction.

□

Claim 25. *Assuming* UGS *satisfies 'MAC simulation' (Definition 10), it holds that* $\mathsf{Hybrid}_2 \overset{\mathrm{s}}{\approx} \mathsf{Hybrid}_3$

Proof. The hybrid experiments Hybrid_2 and Hybrid_3 differ only in how the wire labels (for all circuit input wires and the output wires of multiplication gates) (pk_i, X_i) are created. In Hybrid_2, this label is generated using $\mathcal{S}_{\mathsf{MAC}}$, while in Hybrid_3 these are generated using $\mathsf{UGS.MAC}$ like in the real execution. In the former hybrid, note that the input wires to the multiplication gates can either be outputs of preceding multiplication gates or outputs of addition gates. In the latter case, the label (pk_i, X_i) of the input wire i is generated as a linear combination of outputs of $\mathcal{S}_{\mathsf{MAC}}$, for which it still holds that for the randomness ρ_σ that these values are a member of, $n(\rho_i) \leq T$.

Consider for the sake of contradiction that the output distributions of these hybrids were distinguishable with non-negligible advantage ϵ. Then there must exist at least one circuit input wire $\sigma \in [\mathsf{n} + \mathsf{Q}_\times]$ for which the distributions, $\{\mathsf{pk}_\sigma, X_\sigma\}$ (as in Hybrid_2) can be distinguished from $\{\mathsf{pk}_\sigma, X_\sigma\}$ (as in Hybrid_3) with advantage $\geq \frac{\epsilon}{\mathsf{n}+\mathsf{Q}_\times}$, which is also non-negligible. However, it holds due to 'MAC simulation' (Definition 10) that these distributions are statistically close, thereby yielding a contradiction.

□

Claim 26. *Assuming* UGS *satisfies 'ciphertext simulation' (Definition 10),* $\mathsf{Hybrid}_3 \overset{\mathrm{c}}{\approx} \mathsf{Hybrid}_4$

Proof. The hybrid experiments Hybrid_3 and Hybrid_4 differ only in how all the ciphertexts for the multiplication gates are created. In Hybrid_3, all ciphertexts are generated using $\mathcal{S}_{\mathsf{Enc}}$, while in Hybrid_4 they are created as in the real execution as a composition of $\mathsf{UGS.Compile}$ and $\mathsf{UGS.Rerand}$. In order to show that these distributions are indistinguishable, let us consider the following set of $2\mathsf{Q}_\times + 1$ hybrid experiments:

- Hybrid'_0 : this is distributed exactly like the real execution in Hybrid_4.
- $\forall i \in [2\mathsf{Q}_\times]$, Hybrid'_i : this is an intermediate hybrid experiment where the first i ciphertexts are created using $\mathcal{S}_{\mathsf{Enc}}$ as in Hybrid_3 and the rest are created as in the real execution.

  Note that the last of these hybrids, $\mathsf{Hybrid}'_{2\mathsf{Q}_\times} \equiv \mathsf{Hybrid}_3$.

In the above, each set of adjacent hybrids Hybrid'_{i-1} and Hybrid'_i differ only in how the ith ciphertext is created. Consider for the sake of contradiction that there exists a PPT distinguisher D that can distinguish between Hybrid'_0 and $\mathsf{Hybrid}'_{2\mathsf{Q}_\times}$ with non-negligible advantage ϵ. Then there must exist an $i \in [2\mathsf{Q}_\times]$ such that D can distinguish between Hybrid'_{i-1} and Hybrid'_i with probability $> \frac{\epsilon}{2\mathsf{Q}_\times}$. We show that D can be used in a black-box way by a PPT adversary $\mathcal{A}_{\mathsf{Enc}}$ to distinguish in the 'ciphertext simulation' game (Definition 10), as follows:

- $\mathcal{A}_{\mathsf{Enc}}$ samples garbling randomnesses $\rho_{\mathbf{C}}$ and $\rho'_{\mathbf{C}}$ and computes the garbling following the hybrid experiment given in Algorithm 12, until the $i-1^{th}$ ciphertext is created.
- $\mathcal{A}_{\mathsf{Enc}}$ then gives to the challenger $\mathsf{C}_{\mathsf{Enc}}$ the tuple $(\overline{\rho}'_c \star \rho_c, f(\Delta), \mathsf{pk}_1, \mathsf{pk}_2, \overline{\boldsymbol{x}}, \boldsymbol{X}, f(\Delta)(\boldsymbol{x}), \boldsymbol{x}, f)$ and receives a ciphertext c_b.
- $\mathcal{A}_{\mathsf{Enc}}$ uses c_b as the ith ciphertext completes creating $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ by computing the rest of the ciphertexts as in the real execution.
- Note that, if $b = 0$, $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ is distributed as in Hybrid'_{i-1} and otherwise, $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ is distributed as in Hybrid'_i.
- $\mathcal{A}_{\mathsf{Enc}}$ sends $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$ to D and outputs whatever it outputs.

In this execution, $\mathcal{A}_{\mathsf{Enc}}$ has the same advantage as D, that is non-negligible. However, since UGS satisfies 'ciphertext simulation' (Definition 10), no such $\mathcal{A}_{\mathsf{Enc}}$ can exist, and therefore no such D can exist. This yields a contradiction. □

□

Algorithm 12 Experiment Hybrid_3 for RGS

For $\mathsf{pp} = (g, N, \mathcal{X}, \mathcal{Y}) \in \mathit{supp}(\mathsf{Setup})$, where $\mathcal{Y}$ is a module over a ring $\mathbb{R}$, let $\mathcal{F}_{\mathsf{pp}} \colon \mathcal{X}^2 \to \mathcal{Y}$ be a space of bilinear functions (Definition 7) and $\mathsf{UGS} = (\mathsf{Re}_{\mathsf{pk}}, \mathsf{Re}_{\mathsf{pk\text{-}wire}}, \mathsf{Rerand}, \mathsf{Compile}, \mathsf{MAC}, \mathsf{Eval})$ be an updatable garbling scheme (Definitions 8 and 10). For a circuit $\mathbf{C}$ with input $\boldsymbol{x}^{\mathsf{n}} \in \mathcal{X}^{\mathsf{n}}$, the experiment Hybrid_3 operates as follows:

$\mathsf{Hybrid}_3(\rho_{\mathbf{C}}, \mathbf{C}, \boldsymbol{x}^{\mathsf{n}}) \to (\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

- let $\mathbf{C} = (\mathsf{n}, \mathsf{m}, \mathsf{Q}, \{\sigma_1, \sigma_2, \sigma, \mathsf{op}_g\}_{g \in [\mathsf{Q}]})$ be the circuit
- for each wire $\sigma \in [\mathsf{n} + \mathsf{Q}]$, use $\mathbf{C}, \boldsymbol{x}^{\mathsf{n}}$ to compute the real wire values $\boldsymbol{x}_\sigma$
- let $(\mathsf{pk}^*, \mathsf{pk}_w) \leftarrow \mathsf{pp}$ be default public values for UGS
- sample $\Delta' \leftarrow \mathcal{G}_{\mathsf{pk}}(1^\kappa)$
- for all $g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$, sample $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$ and $\overline{\rho}'_\sigma \leftarrow \mathcal{G}_{\mathsf{MAC}}(1^\kappa, \boldsymbol{m}'_\sigma)$
- for all $\sigma \in [\mathsf{n}]$, sample $\boldsymbol{m}'_\sigma \leftarrow \mathsf{UGS}.\mathcal{G}_{\mathsf{offset}}$ and $\overline{\rho}'_\sigma \leftarrow \mathcal{G}_{\mathsf{MAC}}(1^\kappa, \boldsymbol{m}'_\sigma)$
- parse $(\Delta', \{\overline{\rho}'_\sigma, 1, 1\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}'_\sigma\}_{\sigma \in [\mathsf{n}]}) \star \rho_{\mathbf{C}}$
  as $(\Delta, \{\overline{\rho}_\sigma, \overline{\rho}_{c_1,g}, \overline{\rho}_{c_2,g}\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\overline{\rho}_\sigma\}_{\sigma \in [\mathsf{n}]})$
- set $\mathsf{pk} = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk}}(\Delta, \mathsf{pk}^*)$
- for each $g \in [\mathsf{Q}]$ s.t. $\mathsf{op}_g = \times$, set $\mathsf{pk}_\sigma = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \Delta, \mathsf{pk}_w)$
- for each $\sigma \in [\mathsf{n}]$, set $\mathsf{pk}_\sigma = \mathsf{UGS}.\mathsf{Re}_{\mathsf{pk\text{-}wire}}(\overline{\rho}_\sigma \star \Delta, \mathsf{pk}_w)$
- for all $\sigma \in [\mathsf{n}]$,
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $X_\sigma = \mathsf{UGS}.\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)$
- for all gates $g \in [\mathsf{Q}]$ in topological order, let $\sigma_1, \sigma_2, \sigma \in [\mathsf{n} + \mathsf{Q}]$ be the indices of the input and output wires to op_g respectively
  if $\mathsf{op}_g == +$,
  - set $\mathsf{pk}_\sigma = \mathsf{pk}_{\sigma_1} + \mathsf{pk}_{\sigma_2}$, $\boldsymbol{m}'_\sigma = \boldsymbol{m}'_{\sigma_1} + \boldsymbol{m}'_{\sigma_2}$, $X_\sigma = X_{\sigma_1} + X_{\sigma_2}$, and $\overline{\boldsymbol{x}}_\sigma = \overline{\boldsymbol{x}}_{\sigma_1} + \overline{\boldsymbol{x}}_{\sigma_2}$

  if $\mathsf{op}_g == \times$,
  - $\overline{\boldsymbol{x}}_\sigma = \boldsymbol{x}_\sigma + \mathsf{offset}(\rho_\sigma)$ where $\rho_\sigma = \overline{\rho}_\sigma \star \Delta$
  - $X_\sigma = \mathsf{UGS}.\mathsf{MAC}(\rho_\sigma, \boldsymbol{x}_\sigma)$
  - let $f_1(\widehat{\Delta}) \in \mathcal{F}^*_{\mathsf{pp}}$ be a function s.t.
    $f_1(\widehat{\Delta})(x_1, x_2) = (x_1 - \mathsf{offset}(\rho_{\sigma_1})) \times (x_2 - \mathsf{offset}(\rho_{\sigma_2})) + \mathsf{offset}(\rho_\sigma)$
  - let $f_2(\widehat{\Delta}) \in \mathcal{F}^*_{\mathsf{pp}}$ be a key-dependent function s.t.
    $f_2(\widehat{\Delta})(x_1, x_2) = \mathsf{UGS}.\mathsf{MAC}(\overline{\rho}_\sigma \star \widehat{\Delta}, f_1(\widehat{\Delta})(x_1, x_2))$
  - compute $c'_{1,g} \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, (\overline{\boldsymbol{x}}_{\sigma_1}, \overline{\boldsymbol{x}}_{\sigma_2}), (\boldsymbol{X}_{\sigma_1}, \boldsymbol{X}_{\sigma_2}), \overline{\boldsymbol{x}}_\sigma, (\boldsymbol{x}_{\sigma_1}, \boldsymbol{x}_{\sigma_2}), f_1)$
  - compute $c'_{2,g} \leftarrow \mathcal{S}_{\mathsf{Enc}}(\mathsf{pk}_{\sigma_1}, \mathsf{pk}_{\sigma_2}, (\overline{\boldsymbol{x}}_{\sigma_1}, \overline{\boldsymbol{x}}_{\sigma_2}), (\boldsymbol{X}_{\sigma_1}, \boldsymbol{X}_{\sigma_2}), X_\sigma, (\boldsymbol{x}_{\sigma_1}, \boldsymbol{x}_{\sigma_2}), f_2)$
- set $\boldsymbol{F} = (\mathsf{pk}, \{(c'_{1,g}, c'_{2,g}, \mathsf{pk}_\sigma)\}_{g \in [\mathsf{Q}] \text{ s.t. } \mathsf{op}_g = \times}, \{\mathsf{pk}_\sigma\}_{\sigma \in [\mathsf{n}]})$
- set $\boldsymbol{X} = \{(\overline{\boldsymbol{x}}_\sigma, X_\sigma)\}_{\sigma \in [\mathsf{n}]}$
- set $\boldsymbol{d} = \{\boldsymbol{m}_\sigma\}_{\sigma \in [\mathsf{m}]}$
- return $(\boldsymbol{F}, \boldsymbol{X}, \boldsymbol{d})$

F Proof of Theorem 20

Proof. We show that the Σ-protocol $\mathsf{Sigma} = (P_1, V_1, P_2, V_2)$ in Algorithm 8 satisfies completeness, special soundness and special honest-verifier zero-knowledge (Definition 19) as follows:

Completeness. In the protocol in Algorithm 8, for any statement $(\boldsymbol{F}, \boldsymbol{F'})$ in the language (Equation (1)), the verification algorithm V_2 will, except with negligible probability in the security

parameter κ, always accept on a transcript from an honest prover (P_1, P_2). This is since, if

$$\exists \pi_{\mathsf{in}}, \rho_{\mathbf{C}} \in \mathsf{R_{re}}' \text{ s.t. } (\boldsymbol{F'}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) \leftarrow \mathsf{RGS.Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F})$$

then for $\rho^* \leftarrow \mathsf{RGS.Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C})$, it follows from $\mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F'})_0 = \boldsymbol{F}_{\mathsf{pf}}$ (that holds by definition) and the fact that Rerand is used to apply the group action Upd on the garbling $\boldsymbol{F}$ (Definition 3) that $\mathsf{RGS.Rerand}(\rho^* \star \rho_{\mathbf{C}}, \boldsymbol{F})_0 = \boldsymbol{F}_{\mathsf{pf}}$.

Special soundness. Consider the following PPT algorithm,

$\underline{\mathsf{Extract}}(a, (b, c), (b', c'), x)$:
parse $a = \boldsymbol{F}_{\mathsf{pf}}$, $(b, c) = (0, \rho^*)$, $(b', c') = (1, \rho)$, and $x = (\boldsymbol{F}, \boldsymbol{F'})$
check if $\rho^*, \rho \in \mathit{supp}(\mathsf{RGS.Gen}_{\mathsf{mask}})$
check if $\mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F'})_0 == \boldsymbol{F}_{\mathsf{pf}}$ and $\mathsf{RGS.Rerand}(\rho, \boldsymbol{F})_0 == \boldsymbol{F}_{\mathsf{pf}}$
compute $\rho_{\mathbf{C}} = \rho^{*-1} \star \rho$
return $w = \rho_{\mathbf{C}}$

This algorithm internally checks if its inputs are two accepting transcripts of the Σ-protocol in Algorithm 8 and then computes $\rho_{\mathbf{C}} = \rho^{*-1} \star \rho$ to derive the witness. This is possible since $\star$ is a group operation over $\mathsf{R_{re}}$ (Definition 3) and Rerand is used to apply the group action Upd on the garbling $\boldsymbol{F}$. It should therefore follow from Rerand-correctness of masking that $(\boldsymbol{F'}, \pi_{\mathsf{in}}, \pi_{\mathsf{out}}) \leftarrow \mathsf{RGS.Rerand}(\rho_{\mathbf{C}}, \boldsymbol{F})$.

Special honest-verifier zero-knowledge. Consider the following PPT algorithm Sim,

$\underline{\mathsf{Sim}}(x, b)$:
parse $x = (\boldsymbol{F}, \boldsymbol{F'})$ and $b \in \{0, 1\}$
if $b == 0$, set $c = \rho^* \leftarrow \mathsf{RGS.Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C})$ and $a = \mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F'})_0$
if $b == 1$, set $c = \rho^* \leftarrow \mathsf{RGS.Gen}_{\mathsf{mask}}(1^\kappa, \mathbf{C})$ and $a = \mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F})_0$
return (a, c)

We argue that the distribution $\{a, b, c\}_{b \leftarrow \{0,1\}, (a,c) \leftarrow \mathsf{Sim}(x,b)}$ is statistically close to the transcript (a^*, b^*, c^*) generated by a real execution of the Σ-protocol in Algorithm 8, where the randomness is taken over $b^* \leftarrow \{0, 1\}$ and the random coins of the algorithm P_1. Firstly, note that the challenge b and b^* are sampled identically and uniformly from within $\{0, 1\}$. If $b = 0$, (c, a) is computed exactly as in the real protocol and therefore (a, b, c) in the simulation is distributed identically as the real protocol. If $b = 1$, in the simulation, $c = \rho^*$ is sampled afresh from $\mathsf{RGS.Gen}_{\mathsf{mask}}$, instead of being computed as $\rho_{\mathbf{C}} \star \rho^*$. However, by masking (Definition 3), it holds that the distributions of $\{\rho^*\}$ and $\{\rho^* \star \rho_{\mathbf{C}}\}$ are statistically close and a, in the simulation is computed as $\mathsf{RGS.Rerand}(\rho^*, \boldsymbol{F})_0 = \boldsymbol{F}_{\mathsf{pf}}$, to make the verification algorithm V_2 accept, as it would in the real execution. If there existed a distinguisher for the real and ideal transcripts for the case of $b = 1$, this would imply a distinguisher for the masking property of RGS (Definition 3), which cannot exist by assumption.

□