Silent Circuit Relinearisation:
Sublinear-Size (Boolean and Arithmetic)
Garbled Circuits from DCR

Pierre Meyer[1], Claudio Orlandi[1], Lawrence Roy[1], and Peter Scholl[1]

Aarhus University, Aarhus, Denmark
{pierre.meyer,orlandi,peter.scholl}@cs.au.dk, ldr709@gmail.com

Abstract. We introduce a general template for building garbled circuits with low communication, under the decisional composite residuosity (DCR) assumption. For the case of layered Boolean circuits, we can garble a circuit of size s with communication proportional to $O(s/\log\log s)$ bits, plus an additive factor that is polynomial in the security parameter. For layered arithmetic circuits with B-bounded integer computation, we obtain a similar result: the garbled arithmetic circuit has size $O(s/\log\log s) \cdot (\lambda + \log B)$ bits, where λ is the security parameter. These are the first constructions of general-purpose, garbled circuits with sublinear size, without relying on heavy tools like indistinguishability obfuscation or attribute-based and fully homomorphic encryption.

To achieve these results, our main technical tool is a new construction of a form of homomorphic secret sharing where some of the inputs are semi-private, that is, known to one of the evaluating parties. Through a new relinearisation technique that allows performing arbitrary additions and multiplications on semi-private shares, we build such an HSS scheme that supports evaluating any function of the form $C(x) \cdot C'(y)$, where C is any polynomially-sized circuit applied to the semi-private input y, and C' is a restricted-multiplication (or, NC1) circuit applied to the private input x. This significantly broadens the expressiveness of prior known HSS constructions.

Table of Contents

1 Introduction

Garbled circuits, introduced by Yao [Yao82], are a cryptographic tool used to evaluate a circuit on private inputs. Concretely, a garbling scheme consists of an algorithm Garble, which takes a circuit C and outputs a garbled circuit $\widehat{C}$ together with some input encoding functions K_i, for the i-th input. There is also an evaluation algorithm, Eval, which on input $\widehat{C}$ and the encodings $K_i(x_i)$ for some inputs $x_1, \ldots, x_n$, produces a result $y = C(x_1, \ldots, x_n)$. For security, it is required that the garbled circuit $\widehat{C}$ and garbled inputs $K_i(x_i)$ reveal nothing about the inputs or circuit besides y, and some structural information about C.

Garbled circuits are a valuable tool in secure two-party and multi-party computation with constant round complexity, and also enjoy usage in a range of other cryptographic applications. One limitation of most garbled circuit constructions is their bandwidth complexity. Typically, for Boolean circuits, the garbled circuit $\widehat{C}$ has size $O(\lambda)$ times larger than the number of gates in C, for security parameter λ, which introduces a large bandwidth overhead in applications where a garbled circuit needs to be sent over a network. There has been much effort to reduce this complexity over the years, with the state-of-the-art construction based on symmetric cryptography being the "three halves" technique [RR21]. This obtains a size of $\approx 1.5\lambda$ bits per AND gate, and zero bits per XOR gate in the circuit.

For more specialized circuits with components such as branches or lookup tables, techniques such as stacked garbling [HK20, HK21b] and one-hot garbling [HK21a, HKN24] can be used to obtain improvements over naively expressing the components in terms of XOR and AND gates. However, these do not lead to noticeable asymptotic improvements for general, Boolean circuits.

If garbling an arithmetic circuit instead of a Boolean circuit, approaches based on symmetric cryptography are typically less efficient, with the state-of-the-art having a size of $O(\lambda\ell)$ bits per garbled multiplication gate of ℓ-bit integers. Other approaches based on linearly homomorphic encryption [AIK11, BLLL23] can obtain a similar or better communication complexity. Recently, [MORS24] presented a new approach to garbling arithmetic circuits, using techniques drawn from homomorphic secret sharing [BGI16, OSY21, RS21]. This led to a rate-1 arithmetic garbling scheme, where each multiplication gate can be garbled with a cost of sending one Damgård-Jurik ciphertext, which has size roughly the same as the maximum size of any wire value when evaluating the arithmetic circuit over the integers. Notably, this was the first rate-1 construction relying solely on a group-theoretic (factoring-related) assumption.

Succinct Garbled Circuits: Overcoming the Circuit Size Barrier. Several works have shown how to construct *succinct* garbled circuits, where if C is publicly known, then the size of the garbling $\widehat{C}$ can be *sublinear* in the size of C. This was shown to be possible under the subexponential hardness of the learning with errors assumption [GKP+13] with later improvements in [BGG+14]. In the latter construction, the size of the garbled circuit scales polynomially with the depth of the circuit, but not its size. More recently, this dependence on the depth has been removed by relying on LWE with an additional circular security assumption [HLL23]. However, in addition to making strong assumptions, all of these constructions rely on complex primitives such as fully homomorphic encryption and attribute-based encryption. So far, it has not been known how to achieve sublinear-size garbled circuits without such machinery.

1.1 Our Contributions

In this work, we introduce a new approach to garbling Boolean and arithmetic circuits with sublinear size, under the decisional composite residuosity (DCR) assumption. We obtain the

following main results on the communication complexity of garbling layered circuits[1].

Theorem 1. *Assuming DCR, there exists a garbling scheme for layered Boolean circuits, where the size of the garbled circuit and labels is $O(s/\log\log s + m) + (n + D + 1) \cdot \mathsf{poly}(\lambda)$, for a circuit with size s, depth D, n inputs and m outputs.*

For arithmetic circuits, as in several prior works [BLLL23, MORS24], we consider a model of bounded integer computation, where there is a known bound $B \in \mathbb{N}$ such that, for all supported inputs to the circuit, the magnitude of all wire values during evaluation is bounded by B.

Theorem 2. *Assuming DCR, there exists a garbling scheme for layered arithmetic circuits supporting B-bounded integer computation, where the size of the garbled circuit and labels is $O(s/\log\log s + m)(\lambda + \log B) + (n + D + 1)\mathsf{poly}(\lambda, \log B)$, for a circuit with size s, depth D, n inputs and m outputs.*

We remark that in both theorems, the dependence on the circuit depth D can be removed by making an additional circular security assumption for the Damgård-Jurik encryption scheme.

Main Tool: a Relinearisation Technique for Semi-Private Homomorphic Secret Sharing. Like the recent work of [MORS24], our work builds a garbling scheme using techniques from homomorphic secret sharing, and in fact, we achieve our results by introducing a new *relinearisation* tool that allows to significantly expand the class of functions supported by certain homomorphic secret sharing schemes. Concretely, we use this to build a form of *semi-private* HSS scheme for two servers, where one portion of the inputs to the function being evaluated must be known to one of the servers. Our scheme can support homomorphic evaluation of functions of the form

$$C(\vec{x}) \cdot C_{\mathsf{rm}}(\vec{y})$$

where $\vec{x}$ is the semi-private input known to one server, $\vec{y}$ is the private input, C is any circuit of polynomial size, and C_{rm} is a "restricted multiplication circuit", roughly equivalent to a log-depth circuit. Additionally, our scheme has an *offline/online* property that we leverage to build garbled circuits, where one server's share of the semi-private input $\vec{x}$ can be sampled ahead of time, independently of the input $\vec{x}$.

Most classic HSS constructions [BGI16, OSY21, RS21] only support the evaluation of functions of the form $C_{\mathsf{rm}}(\vec{y})$. One exception is the work of [CMPR23], which implicitly builds a semi-private, offline-online HSS for functions of the form $C_{\mathsf{rm}}(\vec{x}) \cdot C_{\mathsf{rm}}(\vec{y})$, as part of their construction of constrained pseudorandom functions. Our construction allows to significantly expand the class of functions that can be evaluated on the semi-private input, and we believe this result and the relinearisation technique will be useful in other applications beyond garbling.

1.2 Related Work

In recent, prior (but independent) work, two other papers have proposed constructions of garbled circuits with small size.

---

[1] Recall that in a layered circuit, gates can be partitioned into layers L_i, such that any circuit wire goes from a gate in some layer L_i to one in the next layer L_{i+1}.

Garbling with 1 Bit Per Gate [LWYY24]. Liu, Wang, Yang and Yu [LWYY24] show how to garble Boolean circuits with 1 bit per gate, using homomorphic encryption based on either ring-LWE or NTRU, plus a KDM security assumption. Although their assumptions are very different, under the hood, their construction appears similar to our general garbling template when instantiated for boolean circuits, except they use a homomorphic encryption scheme (used to homomorphically evaluate a PRG) instead of homomorphic secret sharing based on DCR, as we do.

Partial Succinct Garbling [ILL24]. Ishai, Li, and Lin [ILL24] also provide novel techniques for succinct garbling schemes. Like ours, their techniques are inspired by the recent work of [MORS24] using HSS to garble arithemtic circuits, but we extend the results in different directions. The work of [ILL24] provides a *partial garbling scheme* [IW14] which allows to garble circuits of the form $C_{\mathsf{priv}}(C_{\mathsf{pub}}(x), y))$ in a way that communication only depends on the *private* circuit C_{priv} (times a polynomial factor in the security parameter) and crucially independent of the size of the *public* circuit C_{pub} and public input x. Such a notion of garbling is clearly related to our notion of semi-private HSS: in partial garbling, some of the inputs are known to the evaluator; in semi-private HSS, some of the inputs are known to one of the parties. There are two significant differences, however: (1) our class of circuits is more restricted, being $C_{\mathsf{priv}}(x) \cdot C_{\mathsf{pub}}(y)$ where C_{priv} is an RMS program; and (2) in HSS, the communication cost is independent of *both* the public and private circuits (except perhaps a factor related to the depth) as well as the output size. This allows HSS to have a greater level of succinctness, so we believe this abstraction can offer advantages over presenting things solely in terms of garbling.

As an application, both works achieve succinct garbling, but for different classes of functions and with different levels of succinctness. [ILL24] achieves *full-succinctness* but only for limited (yet useful) classes of functions while our garbling schemes achieve *limited succinctness* for any layered circuit.

Finally, in terms of techniques, both works rely crucially on a novel algebraic technique for non-interactive computation of authenticated shares based on the DDLog procedure.

2 Technical Overview

Secure Computation with DDLogs. For this technical overview, let's pretend that $\mathsf{Enc}(x) = g^x$ is a secure encryption of x that can be decrypted with key sk (of course, this is not a secure encryption and cannot be decrypted — later, we actually use secure variants of the Damgård-Jurik cryptosystem [DJ01] where the plaintext appears "in the exponent").

Now, following Roy-Singh [RS21], suppose two parties parties have subtractive shares $\langle \mathsf{sk} \rangle$ of the secret key, that is, shares $\langle \mathsf{sk} \rangle_0$ and $\langle \mathsf{sk} \rangle_1$ such that $\mathsf{sk} = \langle \mathsf{sk} \rangle_1 - \langle \mathsf{sk} \rangle_0$. There is a *distributed discrete logarithm* procedure DDLog, which allows them to non-interactively compute *authenticated shares* of x as $\langle \mathsf{sk} \cdot x \rangle = \mathsf{DDLog}((g^x)^{\langle \mathsf{sk} \rangle})$ (that is, party σ, for $\sigma \in \{0, 1\}$, computes the share $\langle \mathsf{sk} \cdot x \rangle_\sigma = \mathsf{DDLog}((g^x)^{\langle \mathsf{sk} \rangle_\sigma})$). Here, authenticated refers to the fact that the shares are multiplied by the secret key of the underlying cryptosystem. Moreover, thanks to the linear property of subtractive shares, it is also the case that if the parties have authenticated shares $\langle \mathsf{sk} \cdot y \rangle$ of some value y then they can compute authenticated shares of xy as $\langle \mathsf{sk} \cdot xy \rangle = \mathsf{DDLog}((g^x)^{\langle \mathsf{sk} \cdot y \rangle})$. This ability to non-interactively perform multiplication between encrypted values and secret-shared values has been used in several previous works to build homomorphic secret sharing (HSS) [BGI16, OSY21, RS21], constrained PRFs [CMPR23], and more. One apparent limitation of this technique is that it only allows to perform multiplications between encrypted values and shared values – this leads to the *restricted multiplication straight-line (RMS)* model of computation, where one can perform addition between any values, but multiplications are restricted to only take one *memory value* input (e.g., the output

of an internal gate in authenticated secret-shared form), and an *input* value (a value which was input directly by one of the parties in encrypted form). Indeed, despite several years of progress in this area, it is unclear how to move beyond this restricted class of programs.

It is indeed very challenging to non-interactively perform multiplications of secret-shared values. This holds even for restricted cases, for instance where one of the values is known to one party. To be concrete, MPC protocols based on *linearly-homomorphic encryption* (LHE) allow a party that knows a plaintext value y and an encrypted value $\mathsf{Enc}(x)$ to non-interactively compute an encryption of $\mathsf{Enc}(xy)$. Note that there is no known analogue for protocols based on secret-sharing: suppose the parties hold a secret sharing $\langle x \rangle$ and one of the two parties know a plaintext value y. There is currently no known way to non-interactively let the parties derive a sharing $\langle xy \rangle$.

Local Multiplication and Relinearisation of Authenticated Shares. The first and main technical insight of our work is that it is possible to do a form of local multiplication on shared values, followed by a "relinearisation" technique, allowing parties to non-interactively compute authenticated shares of the result of a multiplication xy when one of the parties knows x and y.

Suppose the parties have shares $\langle \mathsf{sk} \cdot x \rangle$ and $\langle \mathsf{sk} \cdot y \rangle$, and party P_1 knows x, y. We can view party P_0's shares as evaluations of a linear function in the secret key, given by:

$$\langle \mathsf{sk} \cdot x \rangle_0 = p_x(\mathsf{sk}) := \langle \mathsf{sk} \cdot x \rangle_1 - \mathsf{sk} \cdot x, \quad \langle \mathsf{sk} \cdot y \rangle_0 = p_y(\mathsf{sk}) := \langle \mathsf{sk} \cdot y \rangle_1 - \mathsf{sk} \cdot y$$

Since P_1 knows x, y as well as its shares, then it knows the coefficients of the above polynomials p_x, p_y.

Then, multiplying the polynomials, P_1 can compute the coefficients of the *degree-two* polynomial $p_x \cdot p_y$, given by $\langle \mathsf{sk} \cdot x \rangle_1 \cdot \langle \mathsf{sk} \cdot y \rangle_1 + C \cdot \mathsf{sk} + xy \cdot \mathsf{sk}^2$, where C depends on P_1's shares and the values x, y. Meanwhile, P_0 can compute its evaluation of this polynomial at sk, denoted z. Intuitively, this setup can now be seen as a special form of *degree-two* authenticated secret sharing under the key $(\mathsf{sk}, \mathsf{sk}^2)$. Indeed, so far, this observation is exactly the same as one from recent zero-knowledge proof systems based on vector oblivious linear evaluation [DIO21, YSWW21]. However, these ZK proof protocols all require a step of interaction, in order to "relinearise" the shares back to degree one.

We observe that, with the help of DDLog, this relinearisation can be done entirely non-interactively. First, subtracting P_1's constant coefficient from P_0's evaluation, we get:

$$\langle \mathsf{sk} \cdot x \rangle_1 \cdot \langle \mathsf{sk} \cdot y \rangle_1 - z = -C \cdot \mathsf{sk} - xy \cdot \mathsf{sk}^2 = (-C - xy \cdot \mathsf{sk}) \cdot \mathsf{sk}$$

Viewing the above as an authenticated sharing of $-C - xy \cdot \mathsf{sk}$, notice that if we provide the parties with an encryption of the inverse of the secret key $\mathsf{Enc}(\mathsf{sk}^{-1}) = g^{\frac{1}{\mathsf{sk}}}$, they can use DDLog to "remove" a layer of authentication, obtaining simply a sharing of $-C - xy \cdot \mathsf{sk}$. Finally, since party 0 knows c, it can remove this from its share so that both parties end up with an authenticated sharing $\langle xy \cdot \mathsf{sk} \rangle$ as desired.

This trick is extremely powerful, as it allows us for the first time to compute authenticated shares of arbitrary circuits for the parts of the computation where the inputs are known to one of the parties.

We call this kind of homomorphic secret-sharing where one of the parties knows the plaintext values *semi-private HSS*. We note that there are related notions in the literature for garbling schemes (e.g., privacy-free garbling [FNO15], partial garbling [IW14] , etc. where parts of or the whole input is known to one of the parties), but this is the first such notion in the context of HSS.

6

Offline-Online HSS. Another key technical observation of this work is that the above trick allows us to split the inputs to our computation into two kinds. The first is the "classic" kind, which we refer to as the *offline inputs* (since they need to be known by both parties before the computation starts). As usual, offline inputs are to be provided in encrypted form e.g., to input a value x into the computation the party owning x provides $\mathsf{Enc}(x) = g^x$.

On top of the classic offline inputs, our relinearization trick allows us to support a new type of inputs to the computation which we call *semi-private, online inputs* (as they can be defined adaptively after the computation starts). An online input is an input which is provided directly as an authenticated secret-sharing where the plaintext value is known to one of the parties, that is, an input x is shared as subtractive shares $\langle \mathsf{sk} \cdot x \rangle$. Crucially, as online inputs are just linear sharings, they can be sampled "lazily" where party P_0 defines their share just as a random value, and party P_1's share will be the necessary correction value (together with the plaintext value, which we assume is known to P_1).

Using the semi-private online inputs and the encryption $\mathsf{Enc}(\mathsf{sk}^{-1})$ the parties can non-interactively compute any arithmetic circuit C on their online inputs. The result of this computation can then be combined with a standard HSS evaluation method for restricted multiplication circuits, obtaining shares of $C(\vec{x}) \cdot C_{\mathsf{rm}}(\vec{y})$, where $\vec{y}$ are the (private) offline inputs and C_{rm} is a restricted multiplication circuit.

Avoiding the Circular Security Assumption. One limitation of the above relinearisation technique is that it requires the parties to have an encryption of sk^{-1}. When using Damgård-Jurik encryption, we are not currently able to prove security of this, so need to make an additional "circular security" (or, key-dependent message) assumption. We also give a second relinearisation technique that avoids the need for an extra assumption, and is provable only based on DCR. This technique differs slightly, in that it takes as input sharings $\langle \mathsf{sk} \cdot x \rangle$ and $\langle \mathsf{sk}' \cdot y \rangle$ under two independent secret keys $\mathsf{sk}, \mathsf{sk}'$, and outputs a sharing $\langle \mathsf{sk} \cdot xy \rangle$ under one of the two keys. This introduces some additional challenges when constructing an offline-online HSS, since now, for instance, when evaluating addition gates, we need to ensure that parties' shares of the two input values are authenticated under the same key. To make this work, we use a "key-switching" technique where given encryptions of one key under another, the parties can switch the key under which a memory value is authenticated. Taking care to avoid any circularity issues, we define a chain of secret keys, which the parties progressively switch through as the circuit evaluation proceeds.

A similar approach was also used for the arithmetic garbling construction of [MORS24]. However, unlike [MORS24], we can avoid having to switch keys at every layer of the circuit, since as long as the inputs to two multiplication gates are under different keys $\mathsf{sk}, \mathsf{sk}'$, we don't need to switch to a new key. The number of distinct secret keys (and hence, additional ciphertexts) needed in our construction scales with a kind of "pebbling depth" of the circuit, which in the worst case may equal the multiplicative depth, but can be much smaller in general.

Masking and Sublinear-size Garbling. Using our offline-online HSS, we design a general template for garbling circuits with high-fan-in *wide-gates* and constant-rate. We then instantiate this template in different ways, to obtain our results for Boolean and arithmetic garbling. We now give a brief overview of the Boolean instantiation, referring to section 5.2 for further details.

The high-level idea is that for each wire in the circuit with value $w \in \{0, 1\}$, we want the garbler and evaluator to obtain semi-private, authenticated shares $\langle y \rangle$ and $\langle \mathsf{sk} \cdot y \rangle$ of the masked wire value $y = w \oplus r$ for some secret mask r known to the garbler, where the evaluator knows y. For the circuit input wires, the garbler can already define its authenticated shares ahead of time, thanks to the offline/online property of the HSS. Then, to evaluate a fan-in ℓ

gate that computes some function f, if the masked values for the input wires are $y_1, \ldots, y_\ell$, where $y_i = w_i \oplus r_i$, then the parties want to evaluate the following inside HSS:

$$((r_1, \ldots, r_\ell, r), (y_1, \ldots, y_\ell)) \mapsto f(y_1 \oplus r_1, \ldots, y_\ell \oplus r_\ell) \oplus r \qquad (1)$$

Here, the r_i's are private inputs initially chosen by the garbler, while the y_i's are semi-private, offline/online shared values that will be known to the evaluator. During the garbling phase, the garbler computes its authenticated shares of the gate output, namely, $\langle \mathsf{sk} \cdot (f(w) \oplus r) \rangle_0$ and $\langle f(w) \oplus r \rangle_0$. Then, to preserve the invariant that the evaluator knows the masked value, the garbler includes $\langle f(w) \oplus r \rangle_0$ in the garbled circuit. This way, computation can continue through to the end of the circuit.

One problem with this sketch is that the r_i masks all need to be input into HSS as private input values (i.e. ciphertexts), which would lead to a very high communication cost. To avoid this, we have the garbler choose and input a key for a pseudorandom function, and derive all the r_i's pseudorandomly by evaluating the PRF as part of the HSS computation in eq. (1).

Overall, if the HSS is sufficiently expressive to evaluate eq. (1), for each gate, then the garbled circuit predominantly consists of a single, 1-bit wire mask for each fan-in ℓ gate. We observe that our HSS is powerful enough to evaluate this for any gate whose function f has a polynomially sized truth table. Since a single such truth table can be used to capture a $\log\log(s)$-depth chunk of a fan-in-two circuit of size s, we can convert any layered, size-s circuit with fan-in two gates into a circuit with large, truth table gates of size $s/\log\log s$ [Cou19]. Our scheme has a cost of 1 bit per truth table gate, leading to the result in Theorem 1.

To handle arithmetic circuits (Theorem 2), we generalize the above template to support arbitrary masking functions instead of just XOR. By instantiating this with additive (statistical) masking over the integers, we can build a garbling scheme where the garbler needs to send a single integer mask in order to evaluate a polynomial of degree $\log(s)$. This leads to a similar $\log\log(s)$ reduction in size for a general, layered arithmetic circuit of size s.

3 Preliminaries

Notation. We write $\mathbb{Z}/N\mathbb{Z}$ to mean the ring of integers modulo N, and denote its additive and multiplicative subgroups by $(\mathbb{Z}/N\mathbb{Z})^+, (\mathbb{Z}/N\mathbb{Z})^\times$, respectively. We write $[n]$ to denote the set of integers $\{1, \ldots, n\}$, and $[a, b)$ the set $\{a, a+1, \ldots, b-1\}$. We frequently refer to *subtractive shares* of a secret x, which means there are shares $\langle x \rangle_0, \langle x \rangle_1$ such that $\langle x \rangle_1 - \langle x \rangle_0 = x$, where the shares and subtraction operation are in $\mathbb{Z}$, unless otherwise specified.

RSA modulus generation. Let λ be a security parameter. We define $\mathsf{RSA.Gen}$ to be a polynomial-time algorithm which, on input λ, outputs (N, p, q), where $N = pq$ and p, q are both ℓ-bit primes for some length parameter ℓ chosen to ensure λ-bit security.

3.1 Damgård-Jurik Cryptosystem

The Damgård-Jurik cryptosystem [DJ01] is a generalisation of the Paillier cryptosystem [Pai99] (which, with the notation of fig. 1, can be seen as the special case $\zeta = 1$).

Definition 3 (Decision Composite Residuosity Assumption (DCR), [Pai99]). *Let λ be a security parameter. We say that the* Decision Composite Residuosity (DCR) *problem is hard relative to modulus-sampling algorithm* $\mathsf{RSA.Gen}(1^\lambda)$ *if*

$$\left\{ (N, x) : \begin{array}{l} (N, p, q) \xleftarrow{\$} \mathsf{RSA.Gen} \\ x \xleftarrow{\$} (\mathbb{Z}/N^2\mathbb{Z})^\times \end{array} \right\} \stackrel{c}{\approx} \left\{ (N, x^N \bmod N) : \begin{array}{l} (N, p, q) \xleftarrow{\$} \mathsf{RSA.Gen} \\ x \xleftarrow{\$} (\mathbb{Z}/N^2\mathbb{Z})^\times \end{array} \right\}$$

Theorem 4 (Damgård-Jurik Cryptosystem [DJ01]). *For any choice of the parameter* $\zeta \geq 1$, *the construction of fig. 1 is a CPA-secure linearly homomorphic encryption scheme if and only if the DCR assumption holds.*

LHE Damgård-Jurik Cryptosystem [DJ01]

Requires:

– $\zeta \geq 1$ is a constant defining the plaintext size.
– Functions $\exp\colon (\mathbb{Z}/N^\zeta\mathbb{Z})^+ \to 1 + N(\mathbb{Z}/N^{\zeta+1}\mathbb{Z})$ and $\log\colon 1 + N(\mathbb{Z}/N^{\zeta+1}\mathbb{Z}) \to (\mathbb{Z}/N^\zeta\mathbb{Z})^+$ are defined by the following expressions, as in [RS21]:

$$\exp(x) = \sum_{k=0}^{\zeta} \frac{(Nx)^k}{k!} \qquad \text{and} \qquad \log(1 + Nx) = \sum_{k=1}^{\zeta} \frac{(-N)^{k-1}x^k}{k}$$

DJ.KeyGen(1^λ):

1. Run $(N, p, q) \xleftarrow{\$} \mathsf{RSA.Gen}(1^\lambda)$
2. Compute $\varphi \leftarrow (p-1) \cdot (q-1)$
3. Output (N, φ)

DJ.Enc$_{N,\zeta}(x)$:

1. Sample $r \xleftarrow{\$} (\mathbb{Z}/N^{\zeta+1}\mathbb{Z})^\times$
2. Compute $c \leftarrow r^{N^\zeta} \cdot \exp(x)$
3. Output c

DJ.Eval$_{N,\zeta}(c_1, c_2, \alpha)$:

   // Homomorph. eval. $(x, y) \mapsto \alpha \cdot x + y$

   Compute and output $c \leftarrow c_1^\alpha \cdot c_2$

DJ.Dec$_{N,\zeta,\varphi}(c)$:

1. Compute $\psi = \varphi^{-1}$ in $\mathbb{Z}/N^\zeta\mathbb{Z}$
2. Compute and output $x \leftarrow \psi \cdot \log(c^\varphi)$

Fig. 1: The Damgård-Jurik cryptosystem.

Definition 5 (KDM Security). *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, *whose private-key and message spaces are denoted* $\mathcal{K}$ *and* $\mathcal{M}$ *respectively, is* secure in the presence of key-dependent messages with respect to function class $\mathcal{F} \subseteq \mathcal{K} \to \mathcal{M}$ *(or simply* $\mathcal{F}$-KDM-secure*) if for every* p.p.t. *oracle algorithm* $\mathcal{A}$ *it holds that:*

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},0}}(1^\lambda, \mathsf{pk}) = 1 : (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},1}}(1^\lambda, \mathsf{pk}) = 1 : (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \right] \right| \leq \mathsf{negl}(\lambda) \ ,$$

where oracles $\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},0}$ *and* $\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},1}$ *are defined as in fig. 2.*

Experiment IND-CPA and IND-KDM Security Game

$\underline{\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},0}(f)}$
if $f \notin \mathcal{F}$ **return** $\bot$
else
   $c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(f(\mathsf{sk}))$
   return c

$\underline{\mathcal{O}^{\mathsf{KDM}}_{\mathcal{F},\mathsf{sk},1}(f)}$
if $f \notin \mathcal{F}$ **return** $\bot$
else
   $c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(0^{|f(\mathsf{sk})|})$
   return c
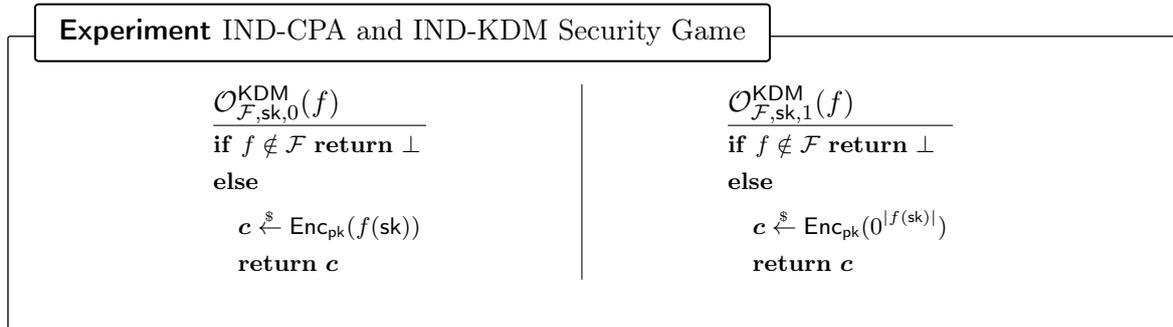
Fig. 2: Oracles used in IND-KDM security (definition 5).

3.2 Damgård-Jurik-ElGamal Cryptosystem

Theorem 6. *Assuming DCR, Damgård–Jurik–ElGamal encryption fig. 3 is a public key encryption scheme satisfying correctness and KDM security for affine functions of the key. Specifically, the following properties hold:*

Correctness: $\mathsf{DJE.Dec_{sk}}(\mathsf{DJE.Enc_{pk}}(x)) = x$, *for any* $(\mathsf{sk}, \mathsf{pk})$ *in the support of* $\mathsf{DJE.KeyGen}$, *and any* $x \in \mathbb{Z}/N^\zeta \mathbb{Z}$.

KDM Security: *Definition 5 for affine functions of the form* $f_{a,b} : \mathsf{sk} \mapsto a \cdot \mathsf{sk} + b$, *where* $a, b \in \mathbb{Z}/N^\zeta \mathbb{Z}$.

Proof. See [BMO$^+$25]. $\qquad\square$

DJE Damgård-Jurik-ElGamal Cryptosystem

Requires:

- $\zeta \geq 1$ is a parameter defining the plaintext size.
- Group isomorphism exp: $(\mathbb{Z}/N^\zeta \mathbb{Z})^+ \to 1 + N(\mathbb{Z}/N^{\zeta+1}\mathbb{Z})$ and its inverse log: $1 + N(\mathbb{Z}/N^{\zeta+1}\mathbb{Z}) \to (\mathbb{Z}/N^\zeta \mathbb{Z})^+$, as defined as in [RS21]:

$$\exp(x) = \sum_{k=0}^{\zeta} \frac{(Nx)^k}{k!} \qquad \text{and} \qquad \log(1 + Nx) = \sum_{k=1}^{\zeta} \frac{(-N)^{k-1}x^k}{k}$$

$\mathsf{DJE.Setup}(1^\lambda)$:

1. Sample $(N, p, q) \xleftarrow{\$} \mathsf{RSA.Gen}(1^\lambda)$ ^{a}
2. Sample $g \xleftarrow{\$} (\mathbb{Z}/N^{\zeta+1}\mathbb{Z})^\times$
3. Output $\mathsf{pp} = (g, N)$

---
^{a} Note that p and q are unused, so N could instead be a CRS.

$\mathsf{DJE.KeyGen}(1^\lambda, \mathsf{pp})$:

1. Parse pp as $\mathsf{pp} = (g, N)$
2. Sample $k \xleftarrow{\$} [0, N)$
3. Compute $h \leftarrow g^{-k}$
4. Output $(\mathsf{sk} = (k, N), \mathsf{pk} = (g, h, N))$

$\mathsf{DJE.Enc_{pk}}(x)$:

1. Parse $\mathsf{pk} = (g, h, N)$
2. Sample $r \xleftarrow{\$} [0, N)$
3. Compute $c_0 \leftarrow g^r$
4. Compute $c_1 \leftarrow h^r \cdot \exp(x)$
5. Output $c = (c_0, c_1)$

$\mathsf{DJE.Dec_{sk}}(c = (c_0, c_1))$:

1. Parse $\mathsf{sk} = (k, N)$
2. Assert $c_0^k \cdot c_1 \equiv 1 \mod N$
3. Output $x \leftarrow \log(c_0^k \cdot c_1)$

Fig. 3: The Damgård-Jurik-ElGamal cryptosystem.

3.3 Distributed Discrete Logarithm

We use two distinct distributed discrete log procedures, for the Damgård-Jurik and Damgård-Jurik-ElGamal cryptosystems. The following two lemmas follow from [RS21, Theorem 18], and were also used in [MORS24].

$\mathsf{DDLog}_N(h \in \mathbb{Z}/N^{\zeta+1}\mathbb{Z})$:

Compute and output $z \leftarrow \log\left(\frac{h}{h \bmod N}\right) \in \mathbb{Z}/N^{\zeta}\mathbb{Z}$

Fig. 4: [RS21]'s distributed discrete logarithm for Damgård-Jurik.

Lemma 7 (Damgård-Jurik Distributed Decryption). *For all* $(N, \varphi) \in$ $\mathsf{Supp}(\mathsf{DJ.KeyGen})$, *for all exponents* $\zeta \geq 1$, *and for all ciphertexts* $c \in (\mathbb{Z}/N^{\zeta+1}\mathbb{Z})^{\times}$ *and shares (over* $\mathbb{Z}$) $\langle x\varphi \rangle_0, \langle x\varphi \rangle_1$ *of some* $x \in \mathbb{Z}$ *times* φ, *we have*

$$\mathsf{DDLog}(c^{\langle x\varphi \rangle_1}) - \mathsf{DDLog}(c^{\langle x\varphi \rangle_0}) \equiv \mathsf{DJ.Dec}_{N,\zeta,\varphi}(c) \cdot x \cdot \varphi \mod N^{\zeta}.$$

Lemma 8 (Damgård-Jurik-ElGamal Distributed Decryption). *If we have shares over* $\mathbb{Z}$ *of* $\langle x \rangle_1 - \langle x \rangle_0 = x$ *and* $\langle k \cdot x \rangle_1 - \langle k \cdot x \rangle_0 = k \cdot x$, *then*

$$\mathsf{DDLog}_N(c_0^{\langle k \cdot x \rangle_1} c_1^{\langle x \rangle_1}) - \mathsf{DDLog}_N(c_0^{\langle k \cdot x \rangle_0} c_1^{\langle x \rangle_0}) \equiv x \cdot y \mod N^{\zeta}$$

always holds, for every choice of plaintext size $\zeta \geq 1$, *key pair* $(\mathsf{sk} = (k, N), \mathsf{pk}) \in$ $\mathsf{Supp}(\mathsf{DJE.KeyGen}(1^{\lambda}))$, *plaintext* $y \in \mathbb{Z}/N^{\zeta}\mathbb{Z}$, *ciphertext* $(c_0, c_1) \in \mathsf{Supp}(\mathsf{DJE.Enc}_{\mathsf{pk}}(y))$, *and scalar* $x \in \mathbb{Z}/N^{\zeta}\mathbb{Z}$.

We also rely on the following, standard lemma stating that shares of some value x modulo M can be lifted to valid shares over the integers, with a failure probability of $|x|/M$.

Lemma 9 (From e.g. [RS21, Lemma 19]). *For all moduli* $M > 1$ *and all modulo* M *shares* $\langle x \rangle_0, \langle x \rangle_1 \in \mathbb{Z}/M\mathbb{Z}$ *of some* $x \in \mathbb{Z}$, *we have*

$$\Pr_{r \xleftarrow{\$} \mathbb{Z}/M\mathbb{Z}}\left[(\langle x \rangle_1 + r) \bmod M - (\langle x \rangle_0 + r) \bmod M = x\right] = \max\left(1 - \frac{|x|}{M}, 0\right).$$

3.4 Models of computation

Arithmetic circuits. Unless otherwise specified, we consider fan-in two arithmetic circuits over the basis $\{+, \times\}$, with a single output gate.

Definition 10 (Arithmetic circuit). An *arithmetic circuit* C (over some ring $\mathcal{R}$) is a rooted, fan-in two, directed acyclic graph whose leaves are labelled by the constant $1_{\mathcal{R}}$ or input variables, whose fan-in two internal nodes are labelled by $+$ or $\times$, and whose fan-in one internal nodes are each labelled by $\cdot_{\alpha}$ for some $\alpha \in \mathcal{R}$.

Each node (or *gate*) of the circuit is associated with an n-variate polynomial (where n is the number of distinct input variables used as labels), defined recursively according to a topological order on the nodes of C (*i.e.* in a *gate-by-gate* fashion):

– a constant gate, labelled $1_{\mathcal{R}}$, computes the constant polynomial $1_{\mathcal{R}}$;
– an input gate, labelled by the ith input variable x_i, computes the polynomial $P_i = (X_1, \ldots, X_n) = X_i$;
– a linear gate, labelled $+$, computes the sum of the polynomials computed by its parents;
– a multiplication gate, labelled $\times$, computes the product of the polynomials computed by its parents;

– a scalar multiplication gate, labelled $\cdot_\alpha$, computes the product of the polynomial computed by its parent with the constant α.

The circuit computes the polynomial P associated with its root, *a.k.a.* the *output gate*. Evaluating C is done by evaluating the associated polynomial function.

We also consider restricted-multiplication circuits [Bar89, Cle90, BGI16, RS21] (sometimes called restricted multiplication straightline programs). This class of circuits is known to contain LOGSPACE (which includes NC1).

Definition 11 (Restricted-multiplication circuit). A *restricted-multiplication circuit* is an arithmetic circuit satisfying the following property: each node labelled $\times$ has at least one predecessor which is a leaf (that is, each multiplication gate has at least one input wire that is a circuit input).

Definition 12 (B-bounded circuit and C-admissible input). Let C be an n-input arithmetic circuit over $\mathbb{Z}$. An input (vector) $x \in \mathbb{Z}^n$ is *admissible for C with respect to bound B*, if the polynomial function computed by *each and every* gate is at most B when evaluated on input x.
If C and/or B is clear from context, we simply say C-admissible, B-admissible, or admissible. If C is an unbounded arithmetic circuit, any input x is C-admissible.

We use the following (non-standard) notion of multiplicative "pebbling" depth. It may be as large as the (standard) multiplicative depth, but is typically much smaller.

Definition 13 (Pebbling depth). Given an arithmetic ciruit, the *pebbling depth* of a gate is defined recursively as follows:

– the pebbling depth of a constant or input gate is 0;
– the pebbling depth of an addition gate whose predecessors have pebbling depth i and j respectively is $\max(i,j)$;
– the pebbling depth of a scalar multiplication gate is the pebbling depth of its parent;
– the pebbling depth of a multiplication gate whose parents have pebbling depth i and j respectively is $\max(i,j,\min(i,j)+1)$ (in other words, if $i=j$ this is $i+1$, and if $i \neq j$ this is $\max(i,j)$).

The *pebbling depth of an arithmetic circuit* is then defined as the pebbling depth of its output gate.

Circuit Classes VP and RMS. We denote by $\mathsf{VP}_n[B]$ the class of B-bounded arithmetic circuits with n inputs and size $\mathsf{poly}(n)$. Similarly, we denote by $\mathsf{RMS}_n[B]$ the class of B-bounded, restricted multiplication arithmetic circuits with n inputs and size $\mathsf{poly}(n)$. When B is clear from context, we simply write VP_n and RMS_n.

Definition 14 (Tensor Product of Circuit Classes). *Let $\mathcal{C}_0$ and $\mathcal{C}_1$ be two circuit classes over a ring $\mathcal{R}$. The* tensor product *of $\mathcal{C}_0$ and $\mathcal{C}_1$, denoted $\mathcal{C}_0 \otimes \mathcal{C}_1$, is defined as the class of all arithmetic circuits composed from some $F_1, \ldots, F_r \in \mathcal{C}_0$ and $G_1, \ldots, G_r \in \mathcal{C}_1$ in the following way: Let L_0 be the union of all input labels from all F_i, and similarly L_1 from all G_i. Then composed circuit has input labels*

$$\{(j,l) \mid j \in \{0,1\}, l \in L_j\}, \qquad \text{and computes} \qquad \sum_{i=1}^{r} F_i \cdot G_i. \tag{2}$$

More specifically, the composed circuit is union of the graphs of the circuits for all F_i and G_i, followed by r multiplication gates and $r-1$ addition gates to compute (2). The integer r associated with this circuit is called the rank.
We write $\mathcal{C}_0 \otimes_r \mathcal{C}_1$ to denote the subset of tensor product circuits with rank at most r.

Remark 15. Note that the size of a tensor product circuit (definition 14) is lower bounded by its rank r. Therefore, when considering classes of circuits with sizes bounded by a polynomial, we can assume that r is bounded by a polynomial.

4 (Semi-Private) Offline-Online HSS

In section 4.1 we provide the definitions of semi-private HSS (definition 16) and its specialisation to offline/online sharing of semi-private inputs (definition 17). In section 4.2 we construct (fig. 6) semi-private offline-online HSS, assuming the KDM security of the Damgård-Jurik cryptosystem, for product circuits of the form $C \cdot C_{\mathsf{rm}}$, where C is an arbitrary circuit applied to the semi-private inputs (known to one party), and C_{rm} is a restricted-multiplication circuit applied to the private inputs. In section 4.3, we present an alternative construction (fig. 9) that assumes only DCR, but achieving a weaker notion of compactness.

4.1 Definition

In HSS with setup, a setup algorithm is used to generate a secret key, used to share any input, in conjunction with a pair of evaluation keys, which can be used to evaluate a function of the shared inputs, obtaining a subtractive sharing of the output. Importantly, each input can be shared independently (as long as they are generated with respect to the same secret key) and need not be batched.

Below, we generalize the standard notion of HSS to allow for a special type of *semi-private input sharing algorithm*, which outputs a pair of shares of an input y such that only party 0's shares are guaranteed to hide y. In addition, we support the usual *private* input sharing algorithm, where each share (individually) reveals no information about the input. Each of the functions supported by the homomorphic evaluation algorithm take inputs partitioned into two sets: a set of n_{priv} private inputs, and a set of $n_{\mathsf{s\text{-}priv}}$ semi-private inputs.

Definition 16 (Semi-Private HSS). Let $\mathcal{R} = \mathcal{R}(\lambda)$ be a ring and $n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}} = \mathsf{poly}(\lambda)$. A (two-party) semi-private HSS scheme (with setup) supporting a class of computations $\mathcal{C} \subseteq (\mathcal{R}^{n_{\mathsf{priv}}} \times \mathcal{R}^{n_{\mathsf{s\text{-}priv}}} \rightarrow \mathcal{R})$ is a tuple of *p.p.t.* algorithms $\mathsf{HSS} = (\mathsf{HSS.Setup}, \mathsf{HSS.Share}_{\mathsf{priv}}, \mathsf{HSS.Share}_{\mathsf{s\text{-}priv}}, \mathsf{HSS.Eval})$ with the following syntax and properties:

- $\mathsf{HSS.Setup}(1^\lambda)$: On input the security parameter 1^λ, the setup algorithm Setup outputs a secret key sk and a pair of evaluation keys $(\mathsf{ek}_0, \mathsf{ek}_1)$.
- $\mathsf{HSS.Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x)$: On input the security parameter 1^λ, secret key sk and a private input $x \in \mathcal{R}$, the input sharing algorithm $\mathsf{Share}_{\mathsf{priv}}$ outputs a pair of input shares $(x^{(0)}, x^{(1)})$.
- $\mathsf{HSS.Share}_{\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y)$: On input the security parameter 1^λ, secret key sk and a semi-private input $y \in \mathcal{R}$, the semi-private input sharing algorithm $\mathsf{Share}_{\mathsf{s\text{-}priv}}$ outputs a pair of input shares $(y^{(0)}, y^{(1)})$.
- $\mathsf{HSS.Eval}(\sigma, \mathsf{ek}_\sigma, (x_i^{(\sigma)})_{i=1}^{n_{\mathsf{priv}}}, (y_i^{(\sigma)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}, C)$: On input a party index σ, an evaluation key ek_σ, a vector of n_{priv} private input shares $(x_i^{(\sigma)})_{i=1}^{n_{\mathsf{priv}}}$, a vector of $n_{\mathsf{s\text{-}priv}}$ semi-private input shares $(y_i^{(\sigma)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}$, and a supported evaluation circuit $C \in \mathcal{C}$, the homomorphic evaluation algorithm Eval returns an output share $y_\sigma \in \mathcal{R}$.

It should satisfy the following properties (where the experiments are detailed in fig. 5):

- **Correctness.** For any sufficiently large $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}$, any C-admissible inputs $((x_i)_{i\in[n_{\mathsf{priv}}]}, (y_i)_{i\in[n_{\mathsf{s\text{-}priv}}]}) \in \mathcal{R}^{n_{\mathsf{priv}}} \times \mathcal{R}^{n_{\mathsf{s\text{-}priv}}}$,

$$\Pr\left[y = C((x_i)_{i\in[n_{\mathsf{priv}}]}, (y_i)_{i\in[n_{\mathsf{s\text{-}priv}}]}) : y \xleftarrow{\$} \mathsf{Exp}_{\mathsf{corr}}^{\mathsf{HSS}}(1^\lambda, (x_i)_{i\in[n_{\mathsf{priv}}]}, (y_i)_{i\in[n_{\mathsf{s\text{-}priv}}]}) \right]$$
$$\geq 1 - \mathsf{negl}(\lambda) \ .$$

– **Security.** For every $\sigma \in \{0,1\}$, there exists a *p.p.t.* algorithm Sim_σ (a simulator), such that for any C-admissible inputs $((x_i)_{i \in [n_{\mathsf{priv}}]}, (y_i)_{i \in [n_{\mathsf{s\text{-}priv}}]}) \in \mathcal{R}^{n_{\mathsf{priv}}} \times \mathcal{R}^{n_{\mathsf{s\text{-}priv}}}$, the output of the experiments $\mathsf{Real}_\sigma^{\mathsf{HSS}}$ and $\mathsf{Ideal}_\sigma^{\mathsf{HSS}}$ (fig. 5) are computationally indistinguishable.

$\mathsf{Exp}_{\mathsf{corr}}^{\mathsf{HSS}}$ / $\mathsf{Exp}_{\mathsf{auth\text{-}corr}}^{\mathsf{HSS}}(1^\lambda, (x_i)_{i \in [n_{\mathsf{priv}}]}, (y_i)_{i \in [n_{\mathsf{s\text{-}priv}}]})$:

$(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda)$

for $i \in [n_{\mathsf{priv}}]$, $(x_i^{(0)}, x_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x_i)$

for $i \in [n_{\mathsf{s\text{-}priv}}]$, $(y_i^{(0)}, y_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{Share}_{\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y_i)$

$z_0 \overset{\$}{\leftarrow} \mathsf{Eval}/\mathsf{AuthEval}(0, \mathsf{ek}_0, (x_i^{(0)})_{i=1}^{n_{\mathsf{priv}}}, (y_i^{(0)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}, C);$

$z_1 \overset{\$}{\leftarrow} \mathsf{Eval}/\mathsf{AuthEval}(1, \mathsf{ek}_1, (x_i^{(1)})_{i=1}^{n_{\mathsf{priv}}}, (y_i^{(1)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}, C);$

return $z \leftarrow z_1 - z_0$

$\mathsf{Real}_\sigma^{\mathsf{HSS}}(1^\lambda)$:

$(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \overset{\$}{\leftarrow} \mathsf{HSS.Setup}(1^\lambda)$

for $i \in [n_{\mathsf{priv}}]$ **do**

$\quad (x_i^{(0)}, x_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x_i)$

for $i \in [n_{\mathsf{s\text{-}priv}}]$ **do**

$\quad (y_i^{(0)}, y_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{Share}_{\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y_i)$

return $(\mathsf{ek}_\sigma, (x_i^{(\sigma)})_{i \in [n_{\mathsf{priv}}]}, (y_i^{(\sigma)})_{i \in [n_{\mathsf{s\text{-}priv}}]})$

$\mathsf{Ideal}_\sigma^{\mathsf{HSS}}(1^\lambda)$:

if $\sigma = 0$ **then**

$\quad$ **return** $\mathsf{Sim}_0(1^\lambda, n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}})$

else

$\quad$ **return** $\mathsf{Sim}_1(1^\lambda, n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}},$

$\qquad\qquad (y_i)_{i \in [n_{\mathsf{s\text{-}priv}}]})$

Fig. 5: Experiments for correctness, authenticated correctness and security in semi-private HSS

In our garbling application, we additionally require a linearity property of the semi-private input sharing algorithm. In particular, we require that the first party's share of a semi-private input y should be generated uniformly and independently of the input; the second party's input share can later be derived as a subtractive share of $y \cdot f(\mathsf{sk})$, where f is some function that maps the secret key to a (vector of) elements of the underlying ring.

Together with this, we define an *authenticated evaluation* property, which requires a special form of homomorphic evaluation algorithm that produces subtractive shares of the output z, as well as $z \cdot f(\mathsf{sk})$.

Definition 17 (Additional Properties of Semi-Private HSS).

– **Linear, offline/online sharing of semi-private inputs.** Let $\mathcal{R} = \mathbb{Z}/M\mathbb{Z}$ for some modulus M. We say the semi-private input sharing algorithm $\mathsf{HSS.Share}_{\mathsf{s\text{-}priv}}$ is *offline/online* if it can be decomposed into two *p.p.t.* algorithms $\mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}$ and $\mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}$, parameterized by a dimension $d \geq 2$ and efficiently computable deterministic function $f : * \to (\mathbb{Z}/M\mathbb{Z})^d$, such that:
  - $\mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda)$, on input the security parameter, outputs a uniformly random share $y^{(0)} \overset{\$}{\leftarrow} (\mathbb{Z}/M\mathbb{Z})^d$
  - $\mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y^{(0)}, y)$, on input the secret key sk, share $y^{(0)}$ and input y, computes and outputs the share $y^{(1)} \leftarrow (y, f(\mathsf{sk}) \cdot y + y^{(0)})$.
  - $\mathsf{HSS.Share}_{\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y)$ generates its output $(y^{(0)}, y^{(1)})$ in the natural way, by first sampling $y^{(0)} \overset{\$}{\leftarrow} \mathsf{Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda)$, and then $y^{(1)} \leftarrow \mathsf{Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y)$.

– **Authenticated evaluation.** Furthermore, for a semi-private HSS scheme with linear

offline-online sharing, we say that it admits *authenticated evaluation*, if there exists an algorithm AuthEval with the same syntax as HSS.Eval, such that:

$$
\Pr\left[
z = (1, f(\mathsf{sk})) \cdot C(\,(x_i)_{i \in [n_{\mathsf{priv}}]}, \quad : z \xleftarrow{\$} \mathsf{Exp}^{\mathsf{HSS}}_{\mathsf{auth\text{-}corr}}(1^\lambda, (x_i)_{i \in [n_{\mathsf{priv}}]}, \atop (y_i)_{i \in [n_{\mathsf{s\text{-}priv}}]}) \quad\quad (y_i)_{i \in [n_{\mathsf{s\text{-}priv}}]})
\right]
$$
$$
\leq 1 - \mathsf{negl}(\lambda)
$$

where $\mathsf{Exp}^{\mathsf{HSS}}_{\mathsf{auth\text{-}corr}}$ is defined in fig. 5.

4.2 Semi-private construction from KDM-security of Damgård-Jurik

Before describing the HSS construction, we introduce our core relinearisation technique.

Core Lemma 1 (Relinearisation, Damgård-Jurik variant). *For all $(N, \varphi) \in \mathsf{Supp}(\mathsf{DJ.KeyGen})$, $c_{\mathsf{inv}} \in \mathsf{Supp}(\mathsf{DJ.Enc}_N(\varphi^{-1} \bmod N^\zeta))$, and $(x, y) \in \mathbb{Z}^2$, if*

1. *$\langle \varphi \cdot x \rangle_1, \langle \varphi \cdot x \rangle_0$ denote shares over $\mathbb{Z}$ of $\varphi \cdot x$ (meaning they are integers satisfying $\langle \varphi \cdot x \rangle_1 - \langle \varphi \cdot x \rangle_0 = \varphi x$)*
2. *$\langle \varphi \cdot y \rangle_1, \langle \varphi \cdot y \rangle_0$ denote shares over $\mathbb{Z}$ of $\varphi \cdot y$*

then $\langle \varphi \cdot xy \rangle_1 - \langle \varphi \cdot xy \rangle_0 \equiv \varphi \cdot xy \bmod N^\zeta$, where $\langle \varphi \cdot xy \rangle_1$ and $\langle \varphi \cdot xy \rangle_0$ are defined as follows:

$$
\begin{cases}
\langle \varphi \cdot xy \rangle_1 \leftarrow -\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_1 \cdot \langle \varphi \cdot y \rangle_1} \right) + (x \cdot \langle \varphi \cdot y \rangle_1 + y \cdot \langle \varphi \cdot x \rangle_1) \\
\langle \varphi \cdot xy \rangle_0 \leftarrow -\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_0 \cdot \langle \varphi \cdot y \rangle_0} \right)
\end{cases}
$$

Proof. Consider the following algebraic identity:

$$
\begin{aligned}
\langle \varphi \cdot x \rangle_0 \cdot \langle \varphi \cdot y \rangle_0 &= (\langle \varphi \cdot x \rangle_1 - \varphi \cdot x) \cdot (\langle \varphi \cdot y \rangle_1 - \varphi \cdot y) \\
&= \varphi^2 \cdot xy + \varphi \cdot (-x \cdot \langle \varphi \cdot y \rangle_1 - y \cdot \langle \varphi \cdot x \rangle_1) + \langle \varphi \cdot x \rangle_1 \cdot \langle \varphi \cdot y \rangle_1
\end{aligned}
$$

After reorganising terms, we get that $\langle \varphi \cdot x \rangle_\sigma \cdot \langle \varphi \cdot y \rangle_\sigma$ is an authenticated share over $\mathbb{Z}$ of $-\varphi \cdot xy + (x \cdot \langle \varphi \cdot y \rangle_1 + y \cdot \langle \varphi \cdot x \rangle_1)$:

$$
\langle \varphi \cdot x \rangle_1 \cdot \langle \varphi \cdot y \rangle_1 - \langle \varphi \cdot x \rangle_0 \cdot \langle \varphi \cdot y \rangle_0 = \varphi \cdot (-\varphi \cdot xy + (x \cdot \langle \varphi \cdot y \rangle_1 + y \cdot \langle \varphi \cdot x \rangle_1))
$$

Therefore, by Lemma 7,

$$
\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_1 \cdot \langle \varphi \cdot y \rangle_1} \right) - \mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_0 \cdot \langle \varphi \cdot y \rangle_0} \right)
$$
$$
\equiv \underbrace{(\mathsf{DJ.Dec}(c_{\mathsf{inv}}) \cdot \varphi)}_{\equiv 1 \bmod N^\zeta} \cdot [-\varphi \cdot xy + (x \cdot \langle \varphi \cdot y \rangle_1 + y \cdot \langle \varphi \cdot x \rangle_1)] \bmod N^\zeta
$$

It follows that

$$
\left(-\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_1 \cdot \langle \varphi \cdot y \rangle_1} \right) + (x \cdot \langle \varphi \cdot y \rangle_1 + y \cdot \langle \varphi \cdot x \rangle_1) \right)
$$
$$
- \left(-\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\langle \varphi \cdot x \rangle_0 \cdot \langle \varphi \cdot y \rangle_0} \right) \right) \equiv \varphi \cdot xy \bmod N^\zeta
$$

$\square$

> **HSS** Semi-private Offline-Online HSS
> (KDM-secure Damgård-Jurik variant)

Requires:

- $\mathsf{DJ} = (\mathsf{DJ.KeyGen}, \mathsf{DJ.Enc}, \mathsf{DJ.Dec})$ is the Damgård-Jurik cryptosystem of fig. 1.
- DDLog is the algorithm of fig. 4.
- $(\mathsf{PRF}_{N,\zeta,s,t,\lambda})_{(N,\zeta,s,t,\lambda)\in\mathbb{N}^5}$ is an $\varepsilon_{\mathsf{PRF}}$-secure family of pseudorandom functions such that $\mathsf{PRF}_{N,\zeta,s,t,\lambda} \colon \{0,1\}^\lambda \times [s+t] \to \mathbb{Z}/N^\zeta\mathbb{Z}$.

$\underline{\mathsf{HSS.Setup}(1^\lambda)} :$

1. $(N,\varphi) \xleftarrow{\$} \mathsf{DJ.KeyGen}(1^\lambda)$
2. $c_{\mathsf{inv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(\varphi^{-1} \bmod N^\zeta)$
3. $\mathsf{sh}_{0,0} \xleftarrow{\$} [0,N]$ // $\langle \varphi \cdot 1 \rangle_0$ (over $\mathbb{Z}$)
4. $\mathsf{sh}_{0,1} \leftarrow \varphi \cdot 1 + \mathsf{sh}_{1,0}$ // $\langle \varphi \cdot 1 \rangle_1$ (over $\mathbb{Z}$)
5. $k_{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^\lambda$
6. $\mathsf{sk} \leftarrow (\varphi, N)$, $\mathsf{ek}_0 \leftarrow (N, c_{\mathsf{inv}}, k_{\mathsf{PRF}}, \mathsf{sh}_{1,0})$, and $\mathsf{ek}_1 = (N, c_{\mathsf{inv}}, k_{\mathsf{PRF}}, \mathsf{sh}_{1,1})$
7. Output $(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1))$

$\underline{\mathsf{HSS.Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x)} :$

1. Parse $\mathsf{sk} = (\varphi, N)$
2. $c_{\mathsf{priv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(x)$
3. Output $(x^{(0)}_{\mathsf{priv}}, x^{(1)}_{\mathsf{priv}}) \leftarrow (c_{\mathsf{priv}}, c_{\mathsf{priv}})$

$\underline{\mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk})} :$

1. $\mathsf{sh}_0 \xleftarrow{\$} \mathbb{Z}/N^\zeta\mathbb{Z}$ // $\langle \varphi \cdot y \rangle_0$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)
2. Output $y^{(0)} \leftarrow \mathsf{sh}_0$

$\underline{\mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y, \mathsf{st})} :$

1. Parse $\mathsf{sk} = (\varphi, N)$
2. $\mathsf{sh}_1 \leftarrow (\varphi \cdot y + y^{(0)}) \bmod N^\zeta$ // $\langle \varphi \cdot y \rangle_1$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)

3. Output $y^{(1)} \leftarrow (y, \mathsf{sh}_1)$

$\underline{\mathsf{HSS.Eval}(\sigma, \mathsf{ek}_\sigma, (x_i^{(\sigma)})_{i\in[n_{\mathsf{priv}}]}, (y_i^{(\sigma)})_{i\in[n_{\mathsf{s\text{-}priv}}]}, C \cdot C_{\mathsf{rm}})} :$

// C is a B-bounded arithmetic circuit, C_{rm} is a B'-bounded restricted-multiplication arithmetic circuit

1. Parse $\mathsf{ek}_\sigma = (N, c_{\mathsf{inv}}, k_{\mathsf{PRF}}, \mathsf{sh}_{1,\sigma})$
2. For $i = 1, \ldots, n_{\mathsf{s\text{-}priv}}$, parse $y_i^{(\sigma)}$ as

$$y_i^{(\sigma)} = \begin{cases} (y, \mathsf{sh}_{i,1}) & \text{if } \sigma = 1 \\ \mathsf{sh}_{i,0} & \text{if } \sigma = 0 \end{cases}$$

   // First, evaluate C on semi-private inputs
3. Parse C as a sequence of gates $g_1, g_2, \ldots, g_{|C|}$, where $g_1, \ldots, g_{n_{\mathsf{s\text{-}priv}}}$ are the input gates.
4. For $i = n_{\mathsf{s\text{-}priv}} + 1, \ldots |C|$:
   // Invariant: $\mathsf{sh}_{i,\sigma}$ stores $\langle \varphi \cdot P_i(\vec{y}) \rangle_\sigma$ (over $\mathbb{Z}$), where P_i is the polynomial computed up to gate g_i

– If g_i is the constant gate 1,

$$\mathsf{sh}_{i,\sigma} \leftarrow \mathsf{sh}_{0,\sigma} \quad // \langle \varphi \cdot 1 \rangle_\sigma$$

– If g_i is an addition gate, whose parents are g_{i_L} and g_{i_R}:

$$\mathsf{sh}_{i,\sigma} \leftarrow \mathsf{sh}_{i_L,\sigma} + \mathsf{sh}_{i_R,\sigma} \quad // \langle \varphi \cdot P_i(\vec{y}) \rangle_\sigma \leftarrow \langle \varphi \cdot (P_{i_L} + P_{i_R})(\vec{y}) \rangle_\sigma$$

– If g_i is a multiplication gate, whose parents are g_{i_L} and g_{i_R}:

$$\mathsf{sh}_{i,\sigma} \leftarrow \begin{cases} -\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\mathsf{sh}_{i_L,\sigma} \cdot \mathsf{sh}_{i_R,\sigma}}\right) + P_{i_L}(\vec{y}) \cdot \mathsf{sh}_{i_R,\sigma} + P_{i_R}(\vec{y}) \cdot \mathsf{sh}_{i_L,\sigma} & \text{if } \sigma = 1 \\ -\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\mathsf{sh}_{i_L,\sigma} \cdot \mathsf{sh}_{i_R,\sigma}}\right) & \text{if } \sigma = 0 \end{cases}$$
$$\mathsf{sh}_{i,\sigma} \leftarrow \mathsf{sh}_{i,\sigma} + \mathsf{PRF}(k_{\mathsf{PRF}}, i) \bmod N^\zeta \quad // \langle \varphi \cdot P_i(\vec{y}) \rangle_\sigma = \langle \varphi \cdot (P_{i_L} P_{i_R})(\vec{y}) \rangle_\sigma$$

– If g_i is a scalar multiplication gate $\cdot_\alpha$, whose parent is g_j:

$$\mathsf{sh}_{i,\sigma} \leftarrow \alpha \cdot \mathsf{sh}_{j,\sigma} \quad // \langle \varphi \cdot \alpha \cdot P_j(\vec{y}) \rangle_\sigma \leftarrow \alpha \cdot \langle \varphi \cdot P_j(\vec{y}) \rangle_\sigma$$

// $\mathsf{sh}_{|C|,\sigma}$ now stores $\langle \varphi \cdot C(\vec{y}) \rangle_\sigma$

5. For $i \in [n_{\mathsf{priv}}]$, parse $x_i^{(\sigma)}$ as ciphertext c_i
   // Next, evaluate C_{rm} on the private inputs $\vec{x}$
6. Parse C_{rm} as a sequence of gates $g'_1, g'_2, \ldots, g'_{|C_{\mathsf{rm}}|}$.
7. For $i = 1, \ldots, |C_{\mathsf{rm}}|$:
   // Invariant: $\mathsf{sh}_{|C|+i,\sigma}$ stores $\langle \varphi \cdot C(\vec{y}) \cdot P_i(\vec{x}) \rangle_\sigma$, where P_i is the polynomial computed up to gate g'_i
   – If g'_i is the constant 1:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{sh}_{|C|,\sigma} \quad // \langle \varphi \cdot C(\vec{y}) \cdot 1 \rangle_\sigma$$

   – If g'_i is the jth input gate:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|,\sigma}}\right) + \mathsf{PRF}(k_{\mathsf{PRF}}, |C|+i) \bmod N^\zeta$$
$$// \langle \varphi \cdot C(\vec{y}) \cdot x_j \rangle_\sigma$$

   – If g'_i is an addition gate, whose parents are g'_{i_L} and g'_{i_R}:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{sh}_{|C|+i_L,\sigma} + \mathsf{sh}_{|C|+i_R,\sigma}$$
$$// \langle \varphi \cdot C(\vec{y}) \cdot (P_{i_L} + P_{i_R})(\vec{x}) \rangle_\sigma$$

   – If g'_i is a scalar multiplication gate $\cdot_\alpha$, whose parent is g'_j:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \alpha \cdot \mathsf{sh}_{|C|+j,\sigma}$$
$$// \langle \varphi \cdot C(\vec{y}) \cdot \alpha \cdot P_j(\vec{x}) \rangle_\sigma \leftarrow \alpha \cdot \langle \varphi \cdot C(\vec{y}) \cdot P_j(\vec{x}) \rangle_\sigma$$

   – If g'_i is a multiplication gate, whose parents are g'_{i_L} and g'_{i_R}, and g'_{i_L} is the jth input:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|+i_R,\sigma}}\right) + \mathsf{PRF}(k_{\mathsf{PRF}}, |C|+i) \bmod N^\zeta$$
$$// \langle \varphi \cdot C(\vec{y}) \cdot x_j \cdot P_{i_R}(\vec{x}) \rangle_\sigma$$

   // $\mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,\sigma}$ stores $\langle \varphi \cdot C(\vec{y}) \cdot C_{\mathsf{rm}}(\vec{x}) \rangle_\sigma$
8. $z_\sigma \leftarrow \mathsf{DDLog}\left((c_{\mathsf{inv}})^{\mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,\sigma}}\right) + \mathsf{PRF}(k_{\mathsf{PRF}}, |C|+|C_{\mathsf{rm}}|) \bmod N^\zeta$ // $\langle C(\vec{y}) \cdot C_{\mathsf{rm}}(\vec{x}) \rangle_\sigma$
9. Output z_σ

$\mathsf{HSS.AuthEval}(\sigma, \mathsf{ek}_\sigma, (x_i^{(\sigma)})_{i \in [n_{\mathsf{priv}}]}, (y_i^{(\sigma)})_{i \in [n_{\mathsf{s\text{-}priv}}]}, C \cdot C_{\mathsf{rm}})$:

1. Perform steps 1-8 of Eval
2. $\tilde{z}_\sigma \leftarrow (z_\sigma, \mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,\sigma})$ // $\langle (1, \varphi) \cdot C(\vec{y}) \cdot C_{\mathsf{rm}}(\vec{x}) \rangle_\sigma$
3. Output $\tilde{z}_\sigma$

Fig. 6: Semi-private offline-online HSS from KDM-security of the Damgård-Jurik encryption scheme.

Theorem 18 (Semi-private offline-online HSS for $\mathsf{RMS} \otimes \mathsf{VP}$ from KDM-security of Damgård-Jurik). *Let λ be a security parameter. Let $\zeta \geq 2$, and assume the KDM-security of the Damgård-Jurik cryptosystem with parameter ζ with respect to the function class $(\varphi, N) \mapsto \varphi^{-1}$; let ℓ_{DCR} denote the bit-length of the corresponding RSA modulus. For every $\varepsilon > 0$, every input sizes $n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}} \in \mathbb{N}^\star$, and every bound $B \leq 2^{(\zeta-1-\varepsilon)\cdot\ell_{\mathsf{DCR}}/2}$, the construction of fig. 6 is a semi-private HSS scheme (definition 16) with linear offline/online sharing and authenticated evaluation (definition 17), supporting the class $\mathsf{RMS}_{n_{\mathsf{s\text{-}priv}}} \otimes_1 \mathsf{VP}_{n_{\mathsf{priv}}}$ (definition 14).*

The proof is deferred to appendix A.1.

Remark 19 (Supporting Rank-r Tensoring). As described, the HSS construction of fig. 6 supports evaluating circuits of the form $C \cdot C_{\mathsf{rm}}$, for $C \in \mathsf{VP}[B]$ and $C_{\mathsf{rm}} \in \mathsf{RMS}[B]$, i.e., circuits in $\mathsf{RMS}[B] \otimes_1 \mathsf{VP}[B]$ as the first inputs go to C_{rm}. This easily extends to evaluating any (higher-rank) circuit in the tensor product class $\mathsf{RMS}[B] \otimes \mathsf{VP}[B]$. Any such circuit computes $\sum_{i=1}^r F_i \cdot G_i$, where $F_i \in \mathsf{VP}, G_i \in \mathsf{RMS}$, so we can evaluate each product separately and then sum up the r sets of shares to obtain the result.

4.3 (Weakly succinct) semi-private construction from DCR

In this section, we present our second HSS construction, which avoids the KDM security assumption.

4.3.1 Relinearisation, without a circular security assumption. We first present a variant of the relinearisation technique, which avoids the need for the key-dependent encryption of φ^{-1}. Unlike our previous method, which multiplies two authenticated sharings $\langle k \cdot x \rangle, \langle k \cdot y \rangle$ to obtain a new sharing $\langle k \cdot xy \rangle$ under the same key, this method is slightly more restrictive: it starts with sharings $\langle k_1 \cdot x \rangle, \langle k_2 \cdot x \rangle$, for independent secret keys k_1, k_2, and outputs a new sharing $\langle k_2 \cdot xy \rangle$.

Core Lemma 2 (Relinearisation, Damgård-Jurik-ElGamal variant). *For all $N, \zeta, x, y \in \mathbb{Z}$, $g \in (\mathbb{Z}/N^\zeta\mathbb{Z})^\times$, and $k_1, k_2 \in [0, N)$, if*

1. *$\langle k_1 \cdot x \rangle_1, \langle k_1 \cdot x \rangle_0$ denote shares over $\mathbb{Z}$ of $k_1 \cdot x$ (meaning they are integers satisfying $\langle k_1 \cdot x \rangle_1 - \langle k_1 \cdot x \rangle_0 = k_1 x$)*
2. *$\langle k_2 \cdot y \rangle_1, \langle k_2 \cdot y \rangle_0$ denote shares over $\mathbb{Z}$ of $k_2 \cdot y$*
3. *c is the matrix $\begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} := \left(\begin{array}{c|c} g & g^{k_2} \\ \hline g^{k_1} & g^{k_1 k_2} \cdot \exp(k_2) \end{array} \right)$*

then $\langle k_2 \cdot xy \rangle_1 - \langle k_2 \cdot xy \rangle_0 \equiv k_2 \cdot xy \bmod N^\zeta$, where $\langle k_2 \cdot xy \rangle_1$ and $\langle k_2 \cdot xy \rangle_0$ are defined as follows:

$$\begin{cases} \langle k_2 \cdot xy \rangle_1 & \leftarrow \mathsf{DDLog}\left(c_{11}^{xy} c_{10}^{-x\langle k_2 \cdot y \rangle_1} c_{01}^{-y\langle k_1 \cdot x \rangle_1} c_{00}^{-\langle k_1 \cdot x \rangle_1 \langle k_2 \cdot y \rangle_1} \right) \\ \langle k_2 \cdot xy \rangle_0 & \leftarrow \mathsf{DDLog}\left(c_{00}^{-\langle k_1 \cdot x \rangle_0 \langle k_2 \cdot y \rangle_0} \right) \end{cases}$$

Proof. Consider the following algebraic identity:

$$
\begin{aligned}
k_1 k_2 \cdot xy &= k_1 \cdot x \cdot \langle k_2 \cdot y \rangle_1 - \overbrace{k_1 \cdot x \cdot \langle k_2 \cdot y \rangle_0}^{= \langle k_1 \cdot x \rangle_1 \cdot \langle k_2 \cdot y \rangle_0 - \langle k_1 \cdot x \rangle_0 \cdot \langle k_2 \cdot y \rangle_0} \\
&= k_1 \cdot x \cdot \langle k_2 \cdot y \rangle_1 - \underbrace{\langle k_1 \cdot x \rangle_1 \cdot \langle k_2 \cdot y \rangle_0}_{= \langle k_1 \cdot x \rangle_1 \cdot \langle k_2 \cdot y \rangle_1 - k_2 \cdot y \cdot \langle k_1 \cdot x \rangle_1} + \langle k_1 \cdot x \rangle_0 \cdot \langle k_2 \cdot y \rangle_0 \\
&= k_1 \cdot x \cdot \langle k_2 \cdot y \rangle_1 - \langle k_1 \cdot x \rangle_1 \cdot \langle k_2 \cdot y \rangle_1 + k_2 \cdot y \cdot \langle k_1 \cdot x \rangle_1 + \langle k_1 \cdot x \rangle_0 \cdot \langle k_2 \cdot y \rangle_0
\end{aligned}
$$

It follows that

$$
\begin{aligned}
c_{11}^{xy} &= g^{k_1 k_2 \cdot xy} \cdot \exp(k_2 \cdot xy) \\
&= c_{10}^{x \langle k_2 \cdot y \rangle_1} \cdot c_{00}^{\langle k_1 \cdot x \rangle_1 \langle k_2 \cdot y \rangle_1} \cdot c_{01}^{y \langle k_1 \cdot x \rangle_1} \cdot c_{00}^{\langle k_1 \cdot x \rangle_0 \langle k_2 \cdot y \rangle_0} \cdot \exp(k_2 \cdot xy)
\end{aligned}
$$

After reorganising terms,

$$
\exp(k_2 \cdot xy) = \left[c_{11}^{xy} c_{10}^{-x \langle k_2 \cdot y \rangle_1} c_{01}^{-y \langle k_1 \cdot x \rangle_1} c_{00}^{-\langle k_1 \cdot x \rangle_1 \langle k_2 \cdot y \rangle_1} \right] / c_{00}^{-\langle k_1 \cdot x \rangle_0 \langle k_2 \cdot y \rangle_0}.
$$

We get then get the desired result by [RS21, Theorem 18]. $\qquad \square$

Regarding security, note that we need to give out the matrix c, which contains $(g, g^{k_1}, g^{k_2}, g^{k_1 k_2} \cdot \exp(k_2))$. While this can be seen as a kind of ElGamal encryption of k_2, where k_2 is also the randomness in the ciphertext, we observe that this is indistinguishable from an encryption of a random value, via an argument similar to that used for proving hardness of the extended-DDH assumption under DCR [HO12]. We prove the following in section A.2.2.

Lemma 20. *Let* $N \xleftarrow{\$} \mathsf{RSA.Gen}(1^\lambda)$, $g \xleftarrow{\$} (\mathbb{Z}/N^{\zeta+1}\mathbb{Z})^\times$, $x, y \xleftarrow{\$} [0, N)$ *and* $r \xleftarrow{\$} [0, N^\zeta)$. *Then, the distributions*

$$
(N, g, g^x, g^y, g^{xy} \cdot \exp(y)) \qquad and \qquad (N, g, g^x, g^y, g^{xy} \cdot \exp(r))
$$

are computationally indistinguishable under the DCR assumption.

4.3.2 Key-switching. Since the circular-secure relinearisation technique changes the underlying key, our HSS construction also needs a *key-switching* mechanism, to ensure that shares input to an addition gate are encoded under the same key.

The KeySwitch algorithm, in fig. 7, uses public group elements $h = g^{-k'}, c = g^{kk'} \cdot \exp(k')$ to convert a sharing $\langle k' \cdot x \rangle$ into $\langle k \cdot x \rangle$, using DDLog. RecKeySwitch, shown in fig. 8, uses the same method with a chain of keys k_i, to iteratively convert the input sharing $\langle k_1 \cdot x \rangle$ into $\langle k_d \cdot x \rangle$. Regarding security, notice that starting with the key k_1, we can apply Lemma 20 to replace the exp term in $c_1 = g^{k_1 k_2} \cdot \exp(k_2)$ with $\exp(r)$ for a random r. In general, in the i-th step of the proof, we will simulate $g^{k_{i-1} k_i}$ using g^{k_i}, and $g^{k_{i+1} k_{i+2}} \exp(k_{i+2})$ using g^{k_i+1}, and replace $\exp(k_{i+2})$ with a random element. We give the complete argument in the security proof of the HSS construction, shown in the next section.

Algorithm Key-Switching

$\mathsf{KeySwitch}_{N,\zeta}(\langle x \rangle_\sigma, \langle k \cdot x \rangle_\sigma, h, c, r)$:   // where $h = g^{-k'}$, $c = g^{kk'} \cdot \exp(k')$

1. Compute $\langle k' \cdot x \rangle_\sigma \leftarrow \mathsf{DDLog}(c^{\langle x \rangle_\sigma} h^{\langle k \cdot x \rangle_\sigma}) + r \bmod N^\zeta$
2. Output $\langle k' \cdot x \rangle_\sigma$

Fig. 7: Key-switching algorithm.

Algorithm Recursive Key-Switching

$\mathsf{RecKeySwitch}_{N,\zeta}(\langle x \rangle_\sigma, \langle k_1 \cdot x \rangle_\sigma, d, (h_i)_{i \in [d]}, (c_i)_{i \in [d-1]}, (r_i)_{i \in [d-1]})$:

// where $h_i = g^{-k_i}$ and $c_i = g^{k_i k_{i+1}} \cdot \exp(k_{i+1})$

  1. For $i = 1, \ldots, d-1$:

      Compute $\langle k_{i+1} \cdot x \rangle_\sigma \leftarrow \mathsf{DDLog}(c_i^{\langle x \rangle_\sigma} h_{i+1}^{\langle k_i \cdot x \rangle_\sigma}) + r_i \bmod N^\zeta$

  2. Output $\langle k_d \cdot x \rangle_\sigma$

Fig. 8: Recursive key-switching algorithm.

Lemma 21 (Correctness of key-switching). *Let $\zeta \geq 1$. For all $x \in \mathbb{Z}$, $(g, N) \in \mathsf{Supp}(\mathsf{DJE.Setup})$, $k, k' \in [0, N)$, if $\langle x \rangle_1$ and $\langle x \rangle_0$ are shares of x over $\mathbb{Z}$, $\langle k \cdot x \rangle_1$ and $\langle k \cdot x \rangle_0$ are shares of $k \cdot x$ over $\mathbb{Z}$, then*

$$\Pr_{r \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}} \left[\langle y \rangle_1 - \langle y \rangle_0 = k' \cdot x \right] \geq 1 - \frac{|x|}{N^{\zeta-1}}$$

where $\langle y \rangle_\sigma \leftarrow \mathsf{KeySwitch}(\langle x \rangle_\sigma, \langle k \cdot x \rangle_\sigma, g^{-k'}, g^{kk'} \exp(k'), r)$, for $\sigma \in \{0, 1\}$.

Moreover, for all $x \in \mathbb{Z}$, $(g, N) \in \mathsf{Supp}(\mathsf{DJE.Setup})$, $d \in \mathbb{N}$, $(k_i)_{i \in [1, d]} \in [0, N)^d$, if $\langle x \rangle_1$ and $\langle x \rangle_0$ are shares of x over $\mathbb{Z}$, $\langle k_1 \cdot x \rangle_1$ and $\langle k_1 \cdot x \rangle_0$ are shares of $k \cdot x$ over $\mathbb{Z}$, then

$$\Pr_{r_1, \ldots, r_{d-1} \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}} \left[\langle y \rangle_1 - \langle y \rangle_0 = k_d \cdot x \right] \geq 1 - \frac{(d-1) \cdot |x|}{N^{\zeta-1}}$$

where $\langle y \rangle_\sigma = \mathsf{RecKeySwitch}_{N,\zeta} \begin{pmatrix} \langle x \rangle_\sigma, \langle k \cdot x \rangle_\sigma, d, (g^{-k_i})_{i \in [d]}, \\ (g^{k_i k_{i+1}} \exp(k_i))_{i \in [d-1]}, (r_i)_{i \in [d-1]} \end{pmatrix}$, for $\sigma \in \{0, 1\}$.

Proof. Let $x \in \mathbb{Z}$, $(g, N) \in \mathsf{Supp}(\mathsf{DJE.Setup})$, $k, k' \in [0, N)$, and $\langle x \rangle_1, \langle x \rangle_0, \langle k \cdot x \rangle_1, \langle k \cdot x \rangle_0 \in \mathbb{Z}$ such that $\langle x \rangle_1 - \langle x \rangle_0 = x$ and $\langle k \cdot x \rangle_1 - \langle k \cdot x \rangle_0 = k \cdot x$.

Defining $h \leftarrow g^{-k'}$ and $c \leftarrow g^{kk'} \cdot \exp(k')$, $c^{\langle x \rangle_1} h^{\langle k \cdot x \rangle_1} / c^{\langle x \rangle_0} h^{\langle k \cdot x \rangle_0} = c^x h^{k \cdot x} = \exp(k' \cdot x)$. It follows by [RS21, theorem 18] that $\mathsf{DDLog}(c^{\langle x \rangle_1} h^{\langle k \cdot x \rangle_1}) - \mathsf{DDLog}(c^{\langle x \rangle_0} h^{\langle k \cdot x \rangle_0}) \equiv k' \cdot x \bmod N^\zeta$. Finally, by Lemma 9, we indeed get

$$\Pr_{r \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}} \Big[\mathsf{KeySwitch}_{N,\zeta}\big(\langle x \rangle_1, \langle k \cdot x \rangle_1, g^{-k'}, g^{kk'} \exp(k'), r \big)$$

$$- \mathsf{KeySwitch}_{N,\zeta}(\langle x \rangle_0, \langle k \cdot x \rangle_0, g^{-k'}, g^{kk'} \exp(k'), r) = k' \cdot x \Big] \geq 1 - \frac{|x|}{N^{\zeta-1}} . \quad (3)$$

Let $x \in \mathbb{Z}$, $(g, N) \in \mathsf{Supp}(\mathsf{DJE.Setup})$, $d \in \mathbb{N}$, $(k_i)_{i \in [1, d]} \in [0, N)^d$, and $\langle x \rangle_1, \langle x \rangle_0, \langle k \cdot x \rangle_1, \langle k \cdot x \rangle_0 \in \mathbb{Z}$ such that $\langle x \rangle_1 - \langle x \rangle_0 = x$ and $\langle k \cdot x \rangle_1 - \langle k \cdot x \rangle_0 = k \cdot x$. For $i \in [d]$, define $h_i \leftarrow g^{-k_i}$ and for $i \in [d-1]$, define $c_i \leftarrow g^{k_i k_{i+1}} \exp(k_{i+1})$.

For $i \in [d-1]$ and $\sigma \in \{0, 1\}$, let $\langle k_{i+1} \cdot x \rangle_\sigma \in \mathbb{Z}$ be defined recursively as $\langle k_{i+1} \cdot x \rangle_\sigma \leftarrow \mathsf{DDLog}((c_i)^{\langle k_i \cdot x \rangle_\sigma} (h_i)^{\langle k_{i+1} \cdot x \rangle_\sigma}) + r_i \bmod N^\zeta$. For $i \in [d]$, consider the random variables $r_1, \ldots, r_n$ all i.i.d. uniform in $\mathbb{Z}/N^\zeta \mathbb{Z}$ consider the event E_i: "$\langle k_i \cdot x \rangle_1 - \langle k_i \cdot x \rangle_0 = k_i \cdot x$". By the first part of this lemma (eq. (3)), $\forall i \in [2, d], \Pr[E_i | E_{i-1}] \geq 1 - |x|/N^{\zeta-1}$. Note that $\Pr[E_1] = 1$ by assumption, and that $\Pr[E_d] \geq \Pr[E_d \wedge E_{d-1}] = \Pr[E_d | E_{d-1}] \cdot \Pr[E_{d-1}] \geq \cdots \geq (\prod_{i=2}^d \Pr[E_i | E_{i-1}]) \cdot \Pr[E_1] \geq (1 - |x|/N^{\zeta-1})^{d-1} \geq 1 - (d-1)|x|/N^{\zeta-1}$. Since E_d is exactly the event whose probability we wished to lower bound, this concludes the proof. $\square$

4.3.3 HSS construction. During homomorphic evaluation of the semi-private part of the circuit, our HSS construction uses key-switching to align secret keys of the input wire shares. Due to this, the size of the evaluation keys, ek_σ, scales linearly with the circuit's pebbling depth. One other difference is that with this version, we can only achieve the regular evaluation procedure HSS.Eval, and not the authenticated evaluation HSS.AuthEval. This makes it slightly more difficult to use in our garbling construction.

HSS Semi-private Offline-Online HSS (Damgård-Jurik-ElGamal variant)

Requires:

- DJE = (DJE.KeyGen, DJE.Enc, DJE.Dec) is the Damgård-Jurik-ElGamal cryptosystem of fig. 3.
- DDLog is the algorithm of fig. 4.
- D is the pebbling depth (definition 13) of the online circuit
- $(\mathsf{PRF}_{N,\zeta,s,t,\lambda})_{(N,\zeta,s,t,\lambda)\in\mathbb{N}^5}$ is an $\varepsilon_{\mathsf{PRF}}$-secure family of pseudorandom functions such that $\mathsf{PRF}_{N,\zeta,s,t,\lambda}\colon ([s]\times[D]\times\{0,1\})\cup[s]\cup([s+t+1,s+t]\times\{0,1\})\to\mathbb{Z}/N^\zeta\mathbb{Z}$.

HSS.Setup(1^λ) :

1. $(g,N)\xleftarrow{\$}\mathsf{DJE.Setup}(1^\lambda)$
2. For $i\in[0,D]$, $(\mathsf{sk}_i=(k_i,N),\mathsf{pk}_i=(g,h_i,N))\xleftarrow{\$}\mathsf{DJE.KeyGen}(1^\lambda,\mathsf{pp})$
3. For $i\in[0,D]$, set $c_i\leftarrow g^{k_ik_{i+1}}\cdot\exp(k_{i+1})$
4. $\mathsf{sh}^{\mathsf{auth}}_{1,0}\xleftarrow{\$}\mathbb{Z}/N^\zeta\mathbb{Z}$ // $\langle k_0\cdot 1\rangle_0$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)
5. $\mathsf{sh}^{\mathsf{auth}}_{1,1}\leftarrow(k_0+\mathsf{sh}^{\mathsf{auth}}_{1,0})\bmod N^\zeta$ // $\langle k_0\cdot 1\rangle_1$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)
6. $k_{\mathsf{PRF}}\xleftarrow{\$}\{0,1\}^\lambda$
7. For $\sigma\in\{0,1\}$, $\mathsf{ek}_\sigma\leftarrow(N,g,(h_i)_{i=1}^D,(c_i)_{i=0}^{D-1},\mathsf{sh}^{\mathsf{auth}}_{1,\sigma},k_{\mathsf{PRF}})$
8. $\mathsf{sk}\leftarrow(g,(k_i)_{i=0}^D,N)$
9. Output $(\mathsf{sk},(\mathsf{ek}_0,\mathsf{ek}_1))$

HSS.Share$_{\mathsf{priv}}(1^\lambda,\mathsf{sk},x)$:

1. Parse $\mathsf{sk}=(g,(k_i)_{i=0}^D,N)$, and compute $\mathsf{pk}_D\leftarrow(g,h_D,N)$
2. $c\xleftarrow{\$}\mathsf{DJE.Enc}_{\mathsf{pk}_D}(x)$
3. $c^{\mathsf{auth}}\xleftarrow{\$}\mathsf{DJE.Enc}_{\mathsf{pk}_D}(k_D\cdot x)$
4. For $\sigma\in\{0,1\}$, set $x^{(\sigma)}\leftarrow(c,c^{\mathsf{auth}})$
5. Output $(x^{(0)},x^{(1)})$

HSS.Share$_{0,\mathsf{s\text{-}priv}}(1^\lambda,\mathsf{sk})$:

1. Parse $\mathsf{sk}=(g,(k_i)_{i=0}^D,N)$
2. $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},0}\xleftarrow{\$}\mathbb{Z}/N^\zeta\mathbb{Z}$ // $\langle k_0\cdot y\rangle_0$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)
3. Set $y^{(0)}\leftarrow\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},0}$ and $\mathsf{st}\leftarrow\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},0}$
4. Output $(y^{(0)},\mathsf{st})$

HSS.Share$_{1,\mathsf{s\text{-}priv}}(1^\lambda,\mathsf{sk},y,\mathsf{st})$:

1. Parse $\mathsf{sk}=(g,(k_i)_{i=0}^D,N)$
2. $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},1}\leftarrow(k_0\cdot y+\mathsf{st})\bmod N^\zeta$ // $\langle k_0\cdot y\rangle_1$ (over $\mathbb{Z}/N^\zeta\mathbb{Z}$)
3. Output $y^{(1)}\leftarrow(y,\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},1})$

HSS.Eval($\sigma,\mathsf{ek}_\sigma,(x_i^{(\sigma)})_{i\in[n_{\mathsf{priv}}]},(y_i^{(\sigma)})_{i\in[n_{\mathsf{s\text{-}priv}}]},(C,C_{\mathsf{rm}}))$:
// C is a B-bounded arithmetic circuit, and C_{rm} is a B'-bounded restricted-multiplication arithmetic circuit.

---

1. Parse $\mathsf{ek}_\sigma \leftarrow (N, g, (h_i)_{i=1}^{D}, (c_i)_{i=0}^{D-1}, \mathsf{sh}_{1,\sigma}^{\mathsf{auth}}, k_{\mathsf{PRF}})$

2. Parse $y_i^{(\sigma)}$ as

$$y_i^{(\sigma)} = \begin{cases} (\mathsf{sh}_{\mathsf{s\text{-}priv},i,1}, \mathsf{sh}_{\mathsf{s\text{-}priv},i,1}^{\mathsf{auth}}, y) & \text{if } \sigma = 1 \\ (\mathsf{sh}_{\mathsf{s\text{-}priv},i,0}, \mathsf{sh}_{\mathsf{s\text{-}priv},i,0}^{\mathsf{auth}}) & \text{if } \sigma = 0 \end{cases}$$

3. Parse C as a sequence of gates $g_1, g_2, \ldots, g_{|C|}$.

4. For $i = 1, \ldots |C|$:

   // Invariant: $\mathsf{sh}_{i,\sigma}^{\mathsf{auth}}$ stores $\langle k_{d_i} \cdot P_i(\vec{y}) \rangle_\sigma$ (over $\mathbb{Z}$), where P_i is the polynomial computed by gate g_i and d_i is the pebbling depth of g_i

   − If g_i is the constant gate 1,

   $$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \leftarrow \mathsf{sh}_{1,\sigma}^{\mathsf{auth}} \;\; // \; \langle k_0 \cdot 1 \rangle_\sigma$$

   − If g_i is the jth input gate,

   $$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \leftarrow \mathsf{sh}_{\mathsf{s\text{-}priv},j,\sigma}^{\mathsf{auth}} \;\; // \; \langle k_0 \cdot y_j \rangle_\sigma$$

   − If g_i is an addition gate at pebbling depth d_i, whose parents are g_{i_L} (at pebbling depth d_L) and g_{i_R} (at pebbling depth d_{i_R}):

   $$\mathsf{sh}_{i,L,\sigma}^{\mathsf{auth},d_i} \leftarrow \mathsf{RecKeySwitch}_{N,\zeta}(\; \mathsf{sh}_{i_L,\sigma}, \mathsf{sh}_{i_L,\sigma}^{\mathsf{auth}}, d_i - d_{i_L},$$
   $$((h_i)_{i \in [d_{i_L}, d_i]}, (c_j)_{j \in [d_{i_L}, d_i - 1]},$$
   $$(\mathsf{PRF}(k_{\mathsf{PRF}}, (i, j, 0)))_{j \in [d_{i_L}, d_i - 1]})$$

   // $\langle k_{\ell_i} \cdot P_{i_L}(\vec{y}) \rangle_\sigma$

   $$\mathsf{sh}_{i,R,\sigma}^{\mathsf{auth},d_i} \leftarrow \mathsf{RecKeySwitch}_{N,\zeta}(\; \mathsf{sh}_{i_R,\sigma}, \mathsf{sh}_{i_R,\sigma}^{\mathsf{auth}}, d_i - d_{i_R},$$
   $$(h_i)_{i \in [d_{i_R}, d_i]}, (c_j)_{j \in [d_{i_R}, d_i - 1]},$$
   $$(\mathsf{PRF}(k_{\mathsf{PRF}}, (i, j, 1)))_{j \in [d_{i_L}, d_i - 1]})$$

   // $\langle k_{\ell_i} \cdot P_{i_R}(\vec{y}) \rangle_\sigma$

   $$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \;\;\; \leftarrow \mathsf{sh}_{i,L,\sigma}^{\mathsf{auth},d_i} + \mathsf{sh}_{i,R,\sigma}^{\mathsf{auth},d_i}$$
   $$// \; \langle k_{\ell_i} \cdot P_i(\vec{y}) \rangle_\sigma \leftarrow \langle k_{\ell_i} \cdot (P_{i_L} + P_{i_R})(\vec{y}) \rangle_\sigma$$

   − If g_i is a multiplication gate at pebbling depth d_i, whose parents are g_{i_L} (at

pebbling depth d_L) and g_{i_R} (at pebbling depth d_{i_R}):

$$\mathsf{sh}_{i_L,\sigma} \leftarrow \begin{cases} P_{i_L}(\vec{y}) & \text{If } \sigma = 1 \\ 0 & \text{If } \sigma = 0 \end{cases}$$

$$\mathsf{sh}_{i_R,\sigma} \leftarrow \begin{cases} P_{i_R}(\vec{y}) & \text{If } \sigma = 1 \\ 0 & \text{If } \sigma = 0 \end{cases}$$

$$\mathsf{sh}_{i,L,\sigma}^{\mathsf{auth},d_i-1} \leftarrow \mathsf{RecKeySwitch}_{N,\zeta}(\,\mathsf{sh}_{i_L,\sigma}, \mathsf{sh}_{i_L,\sigma}^{\mathsf{auth}}, d_i - d_{i_L} - 1,$$
$$(h_j)_{j\in[d_{i_L},d_i-1]}, (c_j)_{j\in[d_{i_L},d_i-2]},$$
$$(\mathsf{PRF}(k_{\mathsf{PRF}}, (i,j,0)))_{j\in[d_{i_L},d_i-1]})$$

$$\mathsf{sh}_{i,R,\sigma}^{\mathsf{auth},d_i} \leftarrow \mathsf{RecKeySwitch}_{N,\zeta}(\,\mathsf{sh}_{i_R,\sigma}, \mathsf{sh}_{i_R,\sigma}^{\mathsf{auth}}, d_i - d_{i_R},$$
$$(h_j)_{j\in[d_{i_R},d_i]}, (c_j)_{j\in[d_{i_R},d_i-1]},$$
$$(\mathsf{PRF}(k_{\mathsf{PRF}}, (i,j,1)))_{j\in[d_{i_L},d_i-1]})$$

$\quad$ // $\langle k_{d_i} \cdot P_{i_R}(\vec{y}) \rangle_\sigma$

$$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \leftarrow \begin{cases} -\mathsf{DDLog}\left((c_{d_i-1})^{P_{i_L}(\vec{y}) \cdot P_{i_R}(\vec{y})} \right. & \text{if } \sigma = 1 \\ \qquad\qquad \cdot (h_i)^{P_{i_L}(\vec{y}) \cdot \mathsf{sh}_{i,R,1}^{\mathsf{auth},d_i}} & \\ \qquad\qquad \cdot (h_{i+1})^{P_{i_R}(\vec{y}) \cdot \mathsf{sh}_{i,L,1}^{\mathsf{auth},d_i-1}} & \\ \qquad\qquad \left. \cdot g^{-\mathsf{sh}_{i,L,1}^{\mathsf{auth},d_i-1} \cdot \mathsf{sh}_{i,R,1}^{\mathsf{auth},d_i}} \right) & \\ -\mathsf{DDLog}\left(g^{-\mathsf{sh}_{i,L,0}^{\mathsf{auth},d_i-1} \cdot \mathsf{sh}_{i,R,0}^{\mathsf{auth},d_i}} \right) & \text{if } \sigma = 0 \end{cases}$$

$$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \leftarrow \mathsf{sh}_{i,\sigma}^{\mathsf{auth}} + \mathsf{PRF}(k_{\mathsf{PRF}}, i) \bmod N^\zeta$$
$\quad$ // $\langle k_{d_i} \cdot P_i(\vec{y}) \rangle_\sigma = \langle k_{d_i} \cdot (P_{i_L} P_{i_R})(\vec{y}) \rangle_\sigma$

- If g_i is a scalar multiplication gate $\cdot_\alpha$, whose parent is g_j:

$$\mathsf{sh}_{i,\sigma}^{\mathsf{auth}} \leftarrow \alpha \cdot \mathsf{sh}_{j,\sigma}^{\mathsf{auth}} \quad // \langle k_{d_i} \cdot \alpha \cdot P_j(\vec{y}) \rangle_\sigma \leftarrow \alpha \cdot \langle k_{d_i} \cdot P_j(\vec{y}) \rangle_\sigma$$

// $\mathsf{sh}_{|C|,\sigma}^{\mathsf{auth}}$ stores $\langle k_D \cdot C(\vec{y}) \rangle_\sigma$
// Now, evaluate C_{rm} on private inputs

5. For $i \in [n_{\mathsf{priv}}]$, parse $x_i^{(\sigma)}$ as $(\mathsf{sh}_{x_i,\sigma}^{\mathsf{auth}}, c_i = (c_{i,0}, c_{i,1}), c_i^{\mathsf{auth}} = (c_{i,0}^{\mathsf{auth}}, c_{i,1}^{\mathsf{auth}}))$
6. Parse C_{rm} as a sequence of gates $g_1', g_2', \dots, g_{|C_{\mathsf{rm}}|}'$.
7. For $i = 1, \dots, |C_{\mathsf{rm}}|$:
$\quad$ // Invariant: $\mathsf{sh}_{|C|+i,\sigma}$ stores $\langle C(\vec{y}) \cdot P_i(\vec{x}) \rangle_\sigma$ and $\mathsf{sh}_{|C|+i,\sigma}^{\mathsf{auth}}$ stores
$\langle k_D \cdot C(\vec{y}) \cdot P_i(\vec{x}_{\mathsf{priv}}) \rangle_\sigma$, where P_i is the polynomial computed by gate g_i'
$\quad$ – If g_i' is the constant 1:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \begin{cases} C(\vec{y}) & \text{if } \sigma = 1 \\ 0 & \text{if } \sigma = 0 \end{cases} // \langle C(\vec{y}) \cdot 1 \rangle_\sigma$$
$$\mathsf{sh}_{|C|+i,\sigma}^{\mathsf{auth}} \leftarrow \mathsf{sh}_{|C|,\sigma}^{\mathsf{auth}} \qquad // \langle k_D \cdot C(\vec{y}) \cdot 1 \rangle_\sigma \leftarrow \langle k_D \cdot C(\vec{y}) \rangle_\sigma$$

$\quad$ – If g_i' is the jth input gate:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{DDLog}\left((c_{j,0})^{\mathsf{sh}_{|C|,\sigma}^{\mathsf{auth}}} (c_{j,1})^{\mathsf{sh}_{|C|,\sigma}} \right) + \mathsf{PRF}_{k_{\mathsf{PRF}}}(|C|+i, 0) \bmod N^\zeta$$
$\quad$ // $\langle C(\vec{y}) \cdot x_j \rangle_\sigma$
$$\mathsf{sh}_{|C|+i,\sigma}^{\mathsf{auth}} \leftarrow \mathsf{DDLog}\left((c_j^{\mathsf{auth}})^{\mathsf{sh}_{|C|,\sigma}^{\mathsf{auth}}} \right) + \mathsf{PRF}(k_{\mathsf{PRF}}, (|C|+i, 1)) \bmod N^\zeta$$
$\quad$ // $\langle k_D \cdot C(\vec{y}) \cdot x_j \rangle_\sigma$

– If g_i' is an addition gate, whose parents are g_{i_L}' and g_{i_R}':

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{sh}_{|C|+i_L,\sigma} + \mathsf{sh}_{|C|+i_R,\sigma}$$
$$// \ \langle C(\vec{y}) \cdot (P_{i_L} + P_{i_R})(\vec{x}) \rangle_\sigma$$
$$\mathsf{sh}^{\mathsf{auth}}_{|C|+i,\sigma} \leftarrow \mathsf{sh}^{\mathsf{auth}}_{|C|+i_L,\sigma} + \mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,\sigma}$$
$$// \ \langle k_D \cdot C(\vec{y}) \cdot (P_{i_L} + P_{i_R})(\vec{x}) \rangle_\sigma$$

– If g_i' is a scalar multiplication gate $\cdot_\alpha$, whose parent is g_j':

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \alpha \cdot \mathsf{sh}_{|C|+j,\sigma}$$
$$//\langle C(\vec{y}) \cdot \alpha \cdot P_j(\vec{x}) \rangle_\sigma \leftarrow \alpha \cdot \langle C(\vec{y}) \cdot P_j(\vec{x}) \rangle_\sigma$$
$$\mathsf{sh}^{\mathsf{auth}}_{|C|+i,\sigma} \leftarrow \alpha \cdot \mathsf{sh}^{\mathsf{auth}}_{|C|+j,\sigma}$$
$$//\langle k_D \cdot C(\vec{y}) \cdot \alpha \cdot P_j(\vec{x}) \rangle_\sigma \leftarrow \alpha \cdot \langle k_D \cdot C(\vec{y}) \cdot P_j(\vec{x}) \rangle_\sigma$$

– If g_i' is a multiplication gate, whose parents are g_{i_L}' and g_{i_R}', and g_{i_L}' is the jth input:

$$\mathsf{sh}_{|C|+i,\sigma} \leftarrow \mathsf{DDLog}\left((c_{j,0})^{\mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,\sigma}}(c_{j,1})^{\mathsf{sh}_{|C|+i_R,\sigma}}\right)$$
$$+\mathsf{PRF}_{k_{\mathsf{PRF}}}(|C| + i, 0) \bmod N^\zeta$$
$$// \ \langle C(\vec{y}) \cdot P_{i_R}(\vec{x}) \rangle_\sigma$$
$$\mathsf{sh}^{\mathsf{auth}}_{|C|+i,\sigma} \leftarrow \mathsf{DDLog}\left((c^{\mathsf{auth}}_{j,0})^{\mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,\sigma}}(c^{\mathsf{auth}}_{j,1})^{\mathsf{sh}_{|C|+i_R,\sigma}}\right)$$
$$+\mathsf{PRF}_{k_{\mathsf{PRF}}}(|C| + i, 1) \bmod N^\zeta$$
$$// \ \langle k_D \cdot C(\vec{y}) \cdot P_{i_R}(\vec{x}) \rangle_\sigma$$

8. $y_\sigma \leftarrow \mathsf{sh}^{\mathsf{auth}}_{|C|+|C_{\mathsf{rm}}|,\sigma}$ $// \ \langle C(\vec{y}) \cdot C_{\mathsf{rm}}(\vec{x}) \rangle_\sigma$
9. Output y_σ

Fig. 9: Semi-private offline-online HSS from KDM-security of the Damgård-Jurik encryption scheme.

Theorem 22 (Semi-private offline-online HSS for $\mathsf{RMS} \otimes \mathsf{VP}$ **from DCR).** *Let λ be a security parameter and assume the DCR assumption with λ_{DCR}-bit RSA modulus. For every $\zeta \geq 1$, every $\varepsilon = \omega(1)$,[2] every input sizes $n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}} \in \mathbb{N}^\star$, and every bound $B \leq 2^{(\zeta-1-\varepsilon)\cdot\lambda_{\mathsf{DCR}}/2}$, the construction of fig. 6 is a semi-private offline-online HSS scheme (definition 16) with linear online-input shares (definition 17), supporting the class $\mathsf{RMS}_{n_{\mathsf{s\text{-}priv}}} \otimes_1 \mathsf{VP}_{n_{\mathsf{priv}}}$ (definition 14).*

As in the previous construction (see remark 19), this also extends to support evaluating the tensor product class $\mathsf{RMS} \otimes \mathsf{VP}$, beyond the rank 1 products in $\mathsf{RMS} \otimes_1 \mathsf{VP}$.

5 Sublinear-Size Garbled Circuits

Our garbling constructions are designed to garble some class of circuits consisting of higher fan-in gates (which we call wide-gates), with constant rate, i.e., a constant size of plaintext values for every gate. Since layered fan-in 2 circuits can be written as smaller wide-gate circuits, with size sublinear in the size of the fan-in 2 circuit, this implies sublinear garbling for fan-in 2 circuits.

---

[2] ε could be chosen as a constant, provided $(1 - 2^{-\varepsilon\cdot\lambda_{\mathsf{DCR}}})^D$ were still negligible, where D is the pebbling depth of the circuit applied to the online inputs.

We base our constant rate garbling schemes on semi-private HSS with offline/online sharing. At a high level, they work as follows: The garbler chooses masks for every wire in the circuit, and gives the evaluator semi-private shares of its masked inputs. These masks and masked values work similarly to the "permute bits' and "color bits" of point-permute garbling, and ensure that the masked values do not leak anything about the plaintext computation. Each (masked) wide-gate is garbled using semi-private HSS, allowing the evaluator to learn its masked output given its masked inputs. This garbled gate also outputs semi-private shares of the masked output, so that the evaluator can evaluate further gates.

In section 5.1 we discuss our requirements on the class of wide-gates. In section 5.2 we present our full construction of garbling of wide-gate circuits. Finally, section 5.3 presents two suitable classes of wide-gates – boolean truth tables and arithmetic multivatiate polynomials – and applies them to fan-in two garbling of boolean and bounded arithmetic circuits.

5.1 Wide-gates and Masking Schemes

Definition 23 (Wide-gates). *Let $\mathcal{X}$ be the plaintext space. A class of wide-gates over $\mathcal{X}$ is a class of functions $\mathcal{G} \subseteq \bigcup_\ell (\mathcal{X}^\ell \to \mathcal{X})$, possibly with multiple different arities ℓ.*

The inputs and outputs of every wide-gate in our garbling scheme must be masked, so that semi-private shares do not leak anything important. We define masking schemes as essentially one-time pads for the plaintext wire values.

Definition 24 (Masking Scheme). *Let λ be a security parameter, and let $(\mathcal{K}_\lambda)_{\lambda \in \mathbb{N}}, (\mathcal{X}_\lambda)_{\lambda \in \mathbb{N}}, (\mathcal{Y}_\lambda)_{\lambda \in \mathbb{N}}$ be infinite families of finite sets. A masking scheme (w.r.t. $(\mathcal{K}_\lambda)_{\lambda \in \mathbb{N}}, (\mathcal{X}_\lambda)_{\lambda \in \mathbb{N}}, (\mathcal{Y}_\lambda)_{\lambda \in \mathbb{N}}$) is a pair of p.p.t. algorithms $(\mathsf{Mask}, \mathsf{UnMask})$ with the following syntax and properties.*

– Mask *takes as input the security parameter 1^λ, a mask $k \in \mathcal{K}$, and a plaintext value $x \in \mathcal{X}$, and outputs a masked value $y \in \mathcal{Y}$; when convenient, we may use $\mathsf{Mask}_k(x)$ as shorthand for $\mathsf{Mask}(1^\lambda, k, x)$.*

– UnMask *takes as input the security parameter 1^λ, a masking key $k \in \mathcal{K}$, and a masked value $y \in \mathcal{Y}$, and outputs a plaintext value $x \in \mathcal{X}$; when convenient, we may use $\mathsf{UnMask}_k(y)$ as shorthand for $\mathsf{UnMask}(1^\lambda, k, y)$.*

– **Correctness.** *There exists a negligible function negl such that*

$$\forall x \in \mathcal{X}_\lambda, \Pr\left[\mathsf{UnMask}_k(\mathsf{Mask}_k(x)) = x\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

– **One-Time Privacy.** *There exists a PPT simulator Sim such that for all $x \in \mathcal{X}_\lambda$, the following distributions are computationally indistinguishable:*

$$\left[y \xleftarrow{\$} \mathsf{Sim}(1^\lambda)\right] \text{ and } \left[y \leftarrow \mathsf{Mask}_k(x) \mid k \xleftarrow{\$} \mathcal{K}\right]$$

Next, while evaluating each gate we must be able to unmask, evaluate some gate, then mask its output, all inside semi-private HSS. We formalize this by saying that the masking scheme and wide-gate class must be admissible for the circuit class supported by the HSS.

Definition 25. *A pair of a masking scheme $(\mathsf{Mask}, \mathsf{UnMask})$ and a class $\mathcal{G}$ of wide-gates is admissible for a circuit class $\mathscr{C}$ if, for all $g \in \mathcal{G}$, the function*

$$(k_1, \ldots, k_\ell, k', y_1, \ldots, y_\ell) \mapsto \mathsf{Mask}_{k'}(g(\mathsf{UnMask}_{k_1}(y_1), \ldots, \mathsf{UnMask}_{k_\ell}(y_\ell))$$

is in $\mathscr{C}$. Additionally, a triplet of masking scheme, wide-gate class, and pseudorandom function F is admissible for $\mathscr{C}$ if, for all $i_1, \ldots, i_\ell, i' \in \mathrm{Dom}(F_k)$,

$$(k, y_1, \ldots, y_\ell) \mapsto \mathsf{Mask}_{F_k(i')}(g(\mathsf{UnMask}_{F_k(i_1)}(y_1), \ldots, \mathsf{UnMask}_{F_k(i_\ell)}(y_\ell)))$$

is in $\mathscr{C}$.

We need this stronger version of admissibility because we need to compress the masking keys using a pseudorandom function, similarly to hybrid encryption. However, for the circuit class $\mathsf{RMS} \otimes \mathsf{VP}$ supported by our offline-online HSS constructions, there's no difference.

Lemma 26. *Let F be a pseudorandom function in NC^1, and let $((\mathsf{Mask}, \mathsf{UnMask}), \mathcal{G})$ be admissible for $\mathsf{RMS} \otimes \mathsf{VP}$. Then $((\mathsf{Mask}, \mathsf{UnMask}), \mathcal{G}, F)$ is admissible for $\mathsf{RMS} \otimes \mathsf{VP}$.*

Proof. The only difference between the two notions of admissibility is setting the mask keys k_j to be PRF evaluations $F_k(i_j)$. This is local to just the RMS side of the tensor product. Since F is in NC^1, it can be evaluated by an RMS program of size $\mathsf{poly}(\lambda)$. One can compose two RMS programs, with the size being at most the product of the two original sizes, so after composing with F the RMS side of the tensor product will still be in RMS. $\qquad\square$

5.2 General Template for Rate-1 Garbling from Offline-Online HSS

Definition 27. *A garbling scheme AGC for a class of n-input m-output arithmetic functions $\mathcal{C} \subseteq (\mathbb{Z}^m)^{\mathbb{Z}^n}$ and bounded input set $\mathcal{B} \subseteq \mathbb{Z}^n$ is a label group $\mathcal{L}$ together with p.p.t. algorithms:*

- $(\widehat{C}, (K_i)_{i \in [n]}) \leftarrow \mathsf{Garble}(C)$: *garble a circuit $C \in \mathcal{C}$ to get a garbled circuit $\widehat{C} \in \{0,1\}^*$ and input wire label encoding functions $K_i : \mathbb{Z} \to \mathcal{L}$ for all $i \in [n]$. Every K_i must be an affine function.*
- $(y_j)_{j \in [m]} \leftarrow \mathsf{Eval}(\widehat{C}, (L_i)_{i \in [n]})$: *evaluate a garbled circuit $\widehat{C}$ on input wire labels $L_i \in \mathcal{L}$ to get outputs $y_j \in \mathbb{Z}$.*

Correctness. AGC *is ε-correct if for all $C \in \mathcal{C}$ and $(x_i)_{i \in [n]} t \in \mathcal{B}$,*

$$\Pr\left[\mathsf{Eval}(\widehat{C}, (L_i)_{i \in [n]}) \neq_{\mathbb{Z}} C(x_1, \ldots, x_n) \;\middle|\; \begin{array}{l} (\widehat{C}, (K_i)_{i \in [n]}) \leftarrow \mathsf{Garble}(C) \\ L_i \leftarrow K_i(x_i) \end{array}\right] \leq \varepsilon.$$

Privacy. AGC *is private if there exists a p.p.t. simulator Sim such that for all $C \in \mathcal{C}$ and $(x_i)_{i \in [n]} \in \mathcal{B}$, the following distributions are indistinguishable.*

$$\left\{ ((L_i)_{i \in [n]}, \widehat{C}) \colon \begin{array}{l} (\widehat{C}, (K_i)_{i \in [n]}) \leftarrow \mathsf{Garble}(C) \\ L_i \leftarrow K_i(x_i) \end{array} \right\} \stackrel{\mathrm{c}}{\approx} \left\{ \mathcal{S}(C, y) \colon y \leftarrow C(x) \right\}.$$

5.2.1 From semi-private HSS. In this section, we provide a general template for rate-1 garbling schemes, given a masking scheme and a semi-private HSS (with linear, offline/online sharing of semi-private inputs and authenticated evaluation) that is expressive enough to capture "unmask, evaluating a single gate, then re-mask".

Our garbling scheme works by having the evaluator iteratively compute a masked representation $\mathsf{Mask}_{R_w}(w)$ of each wire w. The garbled circuit will then contain the mask of the output wire, allowing the evaluator to retrieve its desired output.

We use semi-private homomorphic sharing, where the masked wires are the semi-private inputs and the masks are the private inputs, in order to allow the evaluator to go from masked representations of the input wires of a gate to a masked representation of its output wire. Moreover, the additional properties we require of the HSS scheme (namely linear, offline/online sharing of the semi-private inputs and authenticated evaluation) will guarantee a form of "self reducibility" which will guarantee each gate's output wire is ready to be used as an input wire to the next gate.

At a high level, a semi-private HSS scheme satisfies linear, offline/online sharing of the semi-private inputs if the first share of a semi-private input y consists of $\langle \mathsf{sk} \cdot y \rangle_0$ while the second share consists of $(y, \langle \mathsf{sk} \cdot y \rangle_1)$. After performing authenticated evaluation, the parties hold $\langle f(x,y) \rangle_\sigma$ and $\langle \mathsf{sk} \cdot f(x,y) \rangle_\sigma$ (where x is some private input). If the first party sends $\langle f(x,y) \rangle_0$ to the second, they now hold $\langle \mathsf{sk} \cdot f(x,y) \rangle_0$ and $(f(x,y), \langle \mathsf{sk} \cdot f(x,y) \rangle_1)$. These values are now *almost* consistent with the sharing of a semi-private input, except there is no guarantee that the $\langle \mathsf{sk} \cdot f(x,y) \rangle_\sigma$ are sampled uniformly at random. However, all is well if garbler and evaluator offset these shares by a pseudorandom value.[3]

Combining all of the above ideas yields a garbling scheme, but which is not rate-1:

1. The garbler initially performs the HSS setup, samples the masks for each of the wires, shares the masks (treated as private inputs), samples their offline shares of the masked inputs (treated as public inputs), and samples the PRF key used for share re-randomisation.
2. The garbler then proceeds gate by gate to compute $\langle \mathsf{Mask}_{R_w}(w) \rangle_0$ and $\langle \mathsf{sk} \cdot \mathsf{Mask}_{R_w}(w) \rangle_0$ for each wire w of the circuit. Specifically, if gate f takes as input the wires $w_1, \ldots, w_\ell$ and produces as output the wire w, then the garbler can convert private-input shares of $R_1, \ldots, R_\ell, R$ (the masks of the ℓ inputs to the gate as well as that of its output) and semi-private-input shares of $\mathsf{Mask}_{R_{w_1}}(w_1), \ldots, \mathsf{Mask}_{R_{w_\ell}}(w_\ell)$ into $\langle \mathsf{Mask}_{R_w}(w) \rangle_0, \langle \mathsf{sk} \cdot \mathsf{Mask}_{R_w}(w) \rangle_0$ by homomorphically evaluating

$$(\underbrace{((X_i)_{i=1}^\ell, X)}_{\text{Private Inputs}}, \underbrace{(Y_i)_{i=1}^\ell}_{\substack{\text{Semi-private} \\ \text{Inputs}}}) \mapsto \mathsf{Mask}_X(f(\mathsf{UnMask}_{X_1}(Y_1), \ldots, \mathsf{UnMask}_{X_\ell}(Y_\ell))) \ .$$

3. The garbled circuit is then comprised of the PRF key, the second party's HSS evaluation key, $\langle \mathsf{Mask}_{R_w}(w) \rangle_0$ and the second party's private-input-shares of R_w for each wire w, and the masks for the output wires.
4. An input label for input x is computed as $(\mathsf{Mask}_{R_x}(x), \langle \mathsf{sk} \cdot \mathsf{Mask}_{R_x}(x) \rangle_1)$.
5. At evaluation time, the evaluator proceeds gate-by-gate, computing $\langle \mathsf{Mask}_{R_w}(w) \rangle_1$ and $\langle \mathsf{sk} \cdot \mathsf{Mask}_{R_w}(w) \rangle_1$ analogously to the garbler, and then using $\langle \mathsf{Mask}_{R_w}(w) \rangle_0$ (which is part of the garbled circuit) to reconstruct $\mathsf{Mask}_{R_w}(w)$.
6. Finally, having computed $\mathsf{Mask}_{R_w}(w)$ for the output wire w, the evaluator uses R_w (which is part of the garbled circuit) to retrieve the output value.

The first observation towards making this scheme rate-1 is that the garbled circuit need not contain $\langle \mathsf{Mask}_{R_w}(w) \rangle_0$, only $\langle \mathsf{Mask}_{R_w}(w) \rangle_0 \bmod |\mathcal{Y}|$, where $\mathsf{Mask}_{R_w}(w) \in \mathcal{Y}$ outputs in an interval $[0, |\mathcal{Y}|)$. If the masking scheme is rate-1 then this value is only about as large as the wire value. The second observation is that if all the masks are generated using a PRF F to be evaluated inside the HSS, the garbled circuit only needs to contain the private-input-shares of the bits of the PRF key, not the private-input-shares of each mask. This means the HSS scheme now needs to support the following evaluation:

$$(\underbrace{((X_i)_{i=1}^\lambda, X)}_{\text{Private Inputs}}, \underbrace{(Y_i)_{i=1}^\ell}_{\substack{\text{Semi-private} \\ \text{Inputs}}}) \mapsto \mathsf{Mask}_X \Big(f \big(\mathsf{UnMask}_{F_{X_1 \cdots X_\lambda}(i_1)}(Y_1), \\ \cdots, \\ \mathsf{UnMask}_{F_{X_1 \cdots X_\lambda}(i_\ell)}(Y_\ell) \big) \Big)$$

where $i_1, \ldots, i_\ell$ are the indices of the gate's input wires. This is what admissibility (definition 25) guarantees.

---

[3] Garbler and evaluator can know the PRF key: in the reduction to PRF security, the adversary is the next call to HSS.AuthEval.

Arithmetic Garbling AGC from HSS (for wide-gate circuits)

Requires:

- Bound S on the circuit size.
- A semi-private HSS with modulus M linear, offline/online sharing of semi-private inputs, and supporting authenticated evaluation.
- An admissible triplet for the HSS's circuit class, consisting of:
  1. A class $\mathcal{G}$ of gates over $\mathcal{X}$.
  2. A masking scheme $(\mathcal{K}, \mathcal{X}, \mathcal{Y}, \mathsf{Mask}, \mathsf{UnMask})$, where the space $\mathcal{Y}$ of masked values is an integer interval $[0, |\mathcal{Y}|)$.
  3. A pseudorandom function $F \colon \{0,1\}^\lambda \times [S] \to \mathcal{K}$.
- A pseudorandom function $\mathsf{PRF} \colon \{0,1\}^\lambda \times [S] \to \mathbb{Z}/M\mathbb{Z}$.

The Garbling Scheme (gate-by-gate description):

Initialisation (performed by the garbler).

1. $k_{\mathsf{PRF}} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
2. $(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \overset{\$}{\leftarrow} \mathsf{HSS.Setup}(1^\lambda)$
3. $\vec{r} = (r_1, \ldots, r_\lambda) \overset{\$}{\leftarrow} \{0,1\}^\lambda$
4. For $j \in [\lambda]$, $(r_j^{(0)}, r_j^{(1)}) \overset{\$}{\leftarrow} \mathsf{HSS.Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, r_j)$
5. For each $i \in [n]$, $w_i^{(0)} \overset{\$}{\leftarrow} \mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk})$

HSS circuits (computed by both parties).
Garbling the ith gate $f \in \mathcal{G}$, whose input wires are indexed $i_1, \ldots, i_\ell$.

1. Let C_i be the circuit (in the circuit class of HSS) evaluating the function
$$g_i(\vec{r}, y_1, \ldots, y_\ell) = \mathsf{Mask}_{F_{\vec{r}}(i)}(f(\mathsf{UnMask}_{F_{\vec{r}}(i_1)}(y_1), \ldots, \mathsf{UnMask}_{F_{\vec{r}}(i_\ell)}(y_\ell))).$$

2. $(y_i^{(0)}, \tilde{w}_i^{(0)}) \overset{\$}{\leftarrow} \mathsf{HSS.AuthEval}(0, \mathsf{ek}_0, (r_j^{(0)})_{j \in [\lambda]}, (w_{i_j}^{(0)})_{j \in [\ell]}, C_i)$
3. $w_i^{(0)} \leftarrow \left(\tilde{w}_i^{(0)} + \mathsf{PRF}(k_{\mathsf{PRF}}, i) \right) \bmod M$

Garbler output. The garbler outputs the garbled circuit, comprised of k_{PRF}, ek_1, $r_j^{(1)}$ for $j \in [\lambda]$, $\overline{y}_i^{(0)} \leftarrow (y_i^{(0)} \bmod |\mathcal{Y}|)$ for each of the gates (indexed by i), and $R_j \coloneqq F_{\vec{r}}(j)$ for each output gate index j.

Label generation. For each $i \in [n]$, given input x_i, compute the label $(y_i, w_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, w_i^{(0)}, \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i))$.

Evaluating the ith gate $f \in \mathcal{G}$, whose input wires are indexed $i_1, \ldots, i_\ell$.

1. Parse the garbled circuit as $(k_{\mathsf{PRF}}, \mathsf{ek}_1, (r_j^{(1)})_{j \in [\lambda]}, (\overline{y}_i^{(0)})_i, (R_j)_j)$
2. Define C_i as at garbling time.
3. $(y_i^{(1)}, \tilde{w}_i^{(1)}) \overset{\$}{\leftarrow} \mathsf{HSS.AuthEval}(1, \mathsf{ek}_1, (r_j^{(1)})_{j \in [\lambda]}, (y_{i_j}, w_{i_j}^{(1)})_{j \in [\ell]}, C_i)$
4. $y_i \leftarrow (y_i^{(1)} - \overline{y}_i^{(0)}) \bmod |\mathcal{Y}|$
5. $w_i^{(1)} \leftarrow \tilde{w}_i^{(1)} + \mathsf{PRF}(k_{\mathsf{PRF}}, i) \bmod M$

Evaluator output. For each output gate index j, output $\mathsf{UnMask}_{R_j}(y_j)$.

Fig. 10: Template for (arithmetic or boolean) garbled circuits from semi-private HSS with linear offline/online sharing of semi-private inputs and supporting authenticated evaluation.

Theorem 28 (Template for (arithmetic or boolean) garbled circuits from semi-private HSS with linear offline/online sharing of semi-private inputs and supporting authenticated evaluation). *Let $\mathcal{G}$ be a set of wide-gates over $\mathcal{X}$, $(\mathsf{Mask}, \mathsf{UnMask})$ be a masking scheme with plaintext space $\mathcal{X}$ and masked value space $\mathcal{Y} = [0, |\mathcal{Y}|)$, and F be a PRF. If this triplet is admissible for some circuit class for which we have a semi-private HSS scheme with linear, offline/online sharing of semi-private inputs and supporting authenticated evaluation, then the construction of fig. 10 is an arithmetic garbling scheme for $\mathcal{G}$ wide-gate circuits. For a size s wide-gate circuit with n inputs and m outputs, the size of the garbled circuit and labels is $s \log(|\mathcal{Y}|) + m \log(|\mathcal{K}|) + (n+1)\mathsf{poly}(\lambda)$.*

Proof.

- **Strong correctness.** Let us prove a stronger notion of correctness, namely that with all but negligible probability the following invariants are maintained for every wire:

$$A_i: \qquad "y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)" \qquad \text{(for } i > n \text{ only)}$$
$$B_i: "w_i^{(1)} - w_i^{(0)} = \mathsf{sk} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)" \text{ (for } i \geq 1)$$

  First observe that $\forall i \in [n]$, $\Pr[B_i] = 1$ by definition of $\mathsf{Share}_{0,\mathsf{s\text{-}priv}}$ and $\mathsf{Share}_{1,\mathsf{s\text{-}priv}}$. Now, assume towards a contradiction that there is a non-negligible probability that there exists a wire index $i \geq n$ such that $(y_i \neq \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)) \vee (w_i^{(1)} - w_i^{(0)} \neq \mathsf{sk} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i))$. By a union bound, because there are only polynomially many wires, there exists a wire index $i \geq n$ such that $(y_i \neq \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)) \vee (w_i^{(1)} - w_i^{(0)} \neq \mathsf{sk} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i))$ with non-negligible probability (indeed $1/\mathsf{poly} \leq \Pr[\cup_i(\neg A_i \vee \neg B_i)] \leq \sum_i \Pr[\neg A_i \vee \neg B_i]$). Let $i^\star$ be the smallest such index. It follows from its minimality that there exists a negligible function ε and a non-negligible function δ such that $\Pr[\cap_{i=1}^{i^\star-1}(A_i \cap B_i)] \geq 1 - \varepsilon$ and $\Pr[(\neg A_{i^\star} \vee \neg B_{i^\star})] \geq \delta$. Let $i_1, \ldots, i_\ell$ be the indices of the wires which are inputs to gate $i^\star$. With all but negligible probability, $\forall j \in [\ell], \tilde{w}_{i_j}^{(1)} - \tilde{w}_{i_j}^{(0)} = w_{i_j}^{(1)} - w_{i_j}^{(0)} = \mathsf{sk} \cdot w_{i_j}$ and $y_{i_j} = \mathsf{Mask}_{F_{r_1 \ldots r_\lambda}(i_j)}(w_{i_j})$.

  Consider the following variant of the PRF security game for PRF: (1) the challenger samples k_{PRF}, (2) the adversary gets the PRF evaluation on inputs $[i^\star - 1] \smallsetminus \{i_1, \ldots, i_\ell\}$, (3) the challenger samples $b \xleftarrow{\$} \{0, 1\}$, and sends to the adversary $(\mathsf{PRF}(k_{\mathsf{PRF}}, i_j))_{j \in [\ell]}$ if $b = 0$, and random values instead if $b = 1$, (4) the adversary output b' and wins if $b = b'$. Clearly, any *p.p.t.* adversary must have at most negligible advantage in this game, by PRF security.

  Now, consider the following adversarial strategy: (1) run the garbler's initialisation for the Garbled circuit (which includes sampling sk), with the exception of sampling k_{PRF}, (2) garble the circuit up to gate $i^\star$, using the PRF queries on inputs $[i^\star - 1] \smallsetminus \{i_1, \ldots, i_\ell\}$ for all gates except the the inputs to the $i^\star$-th, (3) receive the challenge $(Y_1, \ldots, Y_\ell)$, (4) output the boolean value of $(y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)) \wedge (\tilde{w}_{i^\star}^{(1)} - \tilde{w}_{i^\star}^{(0)} = \mathsf{sk} \cdot w_{i^\star})$. Observe that if $b = 1$ (*i.e.* the challenges $R_1, \ldots, R_\ell$ are uniformly random) then this adversary must output 1 with all but negligible probability by correctness of $\mathsf{AuthEval}$. Least it constitute an efficient distinguisher, it must therefore also output 1 with all but negligible probability conditioned on $b = 0$. This means that $\Pr[\neg A_i \vee \neg B_i]$ is negligible, which is a contradiction.

- **Privacy.** We must show that the evaluator's view can be simulated using only the circuit

29

and its output. We present a hybrid argument starting from the real garbled circuit and wire labels. The final hybrid will be independent of the inputs x_i, depending only on the circuit C and its outputs z_j, so we can make it be the simulator.

$\mathcal{H}_1$. This is the real world. The evaluator sees the garbled circuit: k_{PRF}, ek_1, $r_j^{(1)}$ for $j \in [\lambda]$, $\overline{y}_i^{(0)} \leftarrow (y_i^{(0)} \bmod |\mathcal{Y}|)$, and $R_j := F_{\vec{r}}(j)$ for each output gate index j. The evaluator also see its wire labels: $(y_i, w_i^{(1)}) \xleftarrow{\$} \mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, w_i^{(0)}, \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i))$.

$\mathcal{H}_2$. By strong correctness, with all but negligible probability every masked wire y_i computed by the evaluator will be correct. I.e., $y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)$ for all i, where w_i is the plaintext value on the wire. Now, y_i is calculated from shares $y_i^{(1)}$ and $\overline{y}_i^{(0)}$, so we rewrite the garbled gates $\overline{y}_i^{(0)}$ in terms of y_i and $y_i^{(1)}$:
$$y_i \leftarrow (y_i^{(1)} - \overline{y}_i^{(0)}) \bmod |\mathcal{Y}|$$
$$\Downarrow$$
$$\overline{y}_i^{(0)} \leftarrow (y_i^{(1)} - y_i) \bmod |\mathcal{Y}|$$

$\mathcal{H}_3$. Notice that the garbler's HSS evaluation key ek_0 is no longer used, since we replaced the garbler's computation of $\overline{y}_i^{(0)}$. Also, the HSS secret key sk is only used to run $\mathsf{HSS.Share}_{\mathsf{priv}}$ and $\mathsf{HSS.Share}_{\mathsf{s\text{-}priv}}$. Therefore, the semi-private security of HSS against party 1 implies that we can simulate ek_1, and the shares $r_j^{(1)}$ and $(y_i, w_i^{(1)})$ from just the semi-private inputs $y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i)$.

$\mathcal{H}_4$. Since we are now using the HSS simulator, the key $\vec{r} = (r_1, \ldots, r_\lambda)$ of F is only used to evaluate F. Therefore, we can replace $F_{\vec{r}}$ by a uniformly random function R from $[S] \rightarrow \mathcal{K}$. This is indistinguishable by the PRF security of F.

$\mathcal{H}_5$. The evaluator's view is now being generated using only the masked values $y_i = \mathsf{Mask}_{R(i)}(w_i)$ of every wire in the circuit. Note that for non-output wires, each $R(i)$ is used exactly once.[4] For non-output wires i, use the one-time privacy (definition 24) of the masking scheme to replace $\mathsf{Mask}_{R(i)}(w_i)$ by a simulation independent of w_i. The adversary's view now only depends on the values of the circuit outputs, so this hybrid is a valid simulator for garbled circuit privacy. □

5.2.2   From semi-private HSS with linear offline/online shares.

Definition 29. *Additionally, a triplet of masking scheme, wide-gate class, and pseudorandom function F is authentication admissible for $\mathscr{C}$ if, for all $i_1, \ldots, i_\ell, i' \in \mathrm{Dom}(F_k)$,*

$$(k, s, y_1, \ldots, y_\ell) \mapsto s \cdot \mathsf{Mask}_{F_k(i')}(g(\mathsf{UnMask}_{F_k(i_1)}(y_1), \ldots, \mathsf{UnMask}_{F_k(i_\ell)}(y_\ell)))$$

is in $\mathscr{C}$.

Remark 30. Similarly to Lemma 26, for our supported HSS circuit class $\mathsf{RMS} \otimes \mathsf{VP}$, authentication admissibility is equivalent to admissibility.

Arithmetic Garbling AGC from HSS (for wide-gate circuits)

Requires:

– Bound S on the circuit size.
– A semi-private HSS with modulus M linear, offline/online sharing of semi-private inputs.
– An admissible (and authentication admissible) triplet for the HSS's circuit class,

---

[4] Recall that for output wires $R(j)$ is also sent to the evaluator to alow it unmask the output.

consisting of:

1. A class $\mathcal{G}$ of gates over $\mathcal{X}$.
2. A masking scheme $(\mathcal{K}, \mathcal{X}, \mathcal{Y}, \mathsf{Mask}, \mathsf{UnMask})$, where the space $\mathcal{Y}$ of masked values is an integer interval $[0, |\mathcal{Y}|)$.
3. A pseudorandom function $F \colon \{0,1\}^\lambda \times [S] \to \mathcal{K}$.
   - A pseudorandom function $\mathsf{PRF} \colon \{0,1\}^\lambda \times [S] \to \mathbb{Z}/M\mathbb{Z}$.

The Garbling Scheme (gate-by-gate description):
We assume that the wires of the circuit (whose depth is D) are ordered in some fixed topological order such that the n input wires are indexed $1, \ldots, n$.

Initialisation (performed by the garbler).

1. $k_{\mathsf{PRF}} \xleftarrow{\$} \{0,1\}^\lambda$
2. For $d \in [0, D], (\mathsf{sk}_d, (\mathsf{ek}_{0,d}, \mathsf{ek}_{1,d})) \xleftarrow{\$} \mathsf{HSS.Setup}(1^\lambda)$
3. $\vec{r} = (r_1, \ldots, r_\lambda) \xleftarrow{\$} \{0,1\}^\lambda$
4. For $d \in [0, D]$ and $j \in [\lambda]$, $(r_{j,d}^{(0)}, r_{j,d}^{(1)}) \xleftarrow{\$} \mathsf{HSS.Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}_d, r_j)$
5. For $d \in [1, D], (s_d^{(0)}, s_d^{(1)}) \xleftarrow{\$} \mathsf{HSS.Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}_{d-1}, \mathsf{sk}_d)$ // Key-switching gadget
6. For each $i \in [n]$ $w_i^{(0)} \xleftarrow{\$} \mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}_0)$ // $\langle \mathsf{sk}_0 \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i) \rangle_0$
7. Let $C_\times$ be the a circuit for the function $(s, y) \mapsto s \cdot y$.
8. For each $i \in [n]$, for $d = 1, \ldots, D$:
$$w_{i,d}^{(0)} \xleftarrow{\$} \left(\mathsf{HSS.Eval}(0, \mathsf{ek}_{0,d-1}, s_d^{(0)}, w_{i,d-1}^{(0)}, \widehat{C}_{\mathsf{id}}) + \mathsf{PRF}(k_{\mathsf{PRF}}, (i, d)) \right) \bmod M$$

Garbling the ith gate f at depth d, whose input wires are indexed $i_1, \ldots, i_\ell$.

1. Let $C_i, \widehat{C}_i$ be the circuits (in the circuit class of HSS) for the functions
$$g_i(\vec{r}, y_1, \ldots, y_\ell) = \mathsf{Mask}_{F_{\vec{r}}(i)}(f(\mathsf{UnMask}_{F_{\vec{r}}(i_1)}(y_1), \ldots, \mathsf{UnMask}_{F_{\vec{r}}(i_\ell)}(y_\ell)))$$
$$\widehat{g}_i(\vec{r}, s, y_1, \ldots, y_\ell) = s \cdot g_i(\vec{r}, y_1, \ldots, y_\ell).$$

2. $y_i^{(0)} \xleftarrow{\$} \mathsf{HSS.Eval}(0, \mathsf{ek}_{0,d}, (r_{j,d}^{(0)})_{j \in [\lambda]}, (w_{i_j,d}^{(0)})_{j \in [\ell]}, C_i)$ // $\langle \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i) \rangle_0$
3. For $d' = d, \ldots, D$:
$$w_{i,d'}^{(0)} \xleftarrow{\$} \Big(\mathsf{HSS.Eval}(0, \mathsf{ek}_{0,d'-1}, ((r_{j,d'-1}^{(0)})_{j \in [\lambda]}, s_{d'}^{(0)}), (w_{i_j,d'-1}^{(0)})_{j \in [\ell]}, \widehat{C}_i)$$
$$+ \mathsf{PRF}(k_{\mathsf{PRF}}, (i, d')) \Big) \bmod M \quad // \langle \mathsf{sk}_{d'} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i) \rangle_0$$

Garbler output. The garbler outputs the garbled circuit, comprised of k_{PRF}, $\mathsf{ek}_{1,d}$ for $d \in [0, D]$, $r_j^{(1)}$ for $j \in [\lambda]$, $s_d^{(1)}$ for $d \in [1, D]$, $\overline{y}_i^{(0)} \leftarrow (y_i^{(0)} \bmod |\mathcal{Y}|)$ for each of the gates (indexed by i), and $R_j := F_{\vec{r}}(j)$ for each output gate index j.

Label generation. For each $i \in [n]$, given input $x_i \in [0, |\mathcal{Y}|)$, compute $y_i \leftarrow \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i)$ and $w_i^{(1)} \xleftarrow{\$} \mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}_0, w_i^{(0)}, \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i))$, and output the label $(y_i, w_i^{(1)})$.

Evaluating the ith input. For $d = 1, \ldots, D$:
$$w_{i,d}^{(1)} \xleftarrow{\$} \left(\mathsf{HSS.Eval}(1, \mathsf{ek}_{1,d-1}, s_d^{(1)}, (y_i, w_{i,d-1}^{(1)}), \widehat{C}_{\mathsf{id}}) + \mathsf{PRF}(k_{\mathsf{PRF}}, (i, d)) \right) \bmod M$$

Evaluating the ith gate f, whose input wires are indexed $i_1, \ldots, i_\ell$.

1. Define C_i and $\widehat{C}_i$ as at garbling time.
2. $y_i^{(1)} \overset{\$}{\leftarrow} \mathsf{HSS.Eval}(1, \mathsf{ek}_{1,d}, (r_{j,d}^{(1)})_{j \in [\lambda]}, (y_{i_j}, w_{i_j,d}^{(1)})_{j \in [\ell]}, C_i)$ $//\ \langle \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i) \rangle_1$
3. $y_i \leftarrow (y_i^{(1)} - \overline{y}_i^{(0)}) \bmod |\mathcal{Y}|$
4. For $d' = d, \dots, D$:

$$w_{i,d'}^{(1)} \overset{\$}{\leftarrow} \Big(\mathsf{HSS.Eval}(1, \mathsf{ek}_{1,d'-1}, ((r_{j,d'-1}^{(1)})_{j \in [\lambda]}, s_{d'}^{(1)}), (y_{i_j}, w_{i_j,d'-1}^{(1)})_{j \in [\ell]}, \widehat{C}_i)$$

$$+ \mathsf{PRF}(k_{\mathsf{PRF}}, (i, d')) \Big) \bmod M \quad //\ \langle \mathsf{sk}_{d'} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i) \rangle_1$$

Evaluator output. For each output gate index j, output $\mathsf{UnMask}_{R_j}(y_j)$.

Fig. 11: Template for (arithmetic or boolean) garbled circuits from semi-private HSS with linear offline/online sharing of semi-private inputs.

Theorem 31 (Template for (arithmetic or boolean) garbled circuits from semi-private HSS). *Let $\mathcal{G}$ be a set of wide-gates over $\mathcal{X}$, $(\mathsf{Mask}, \mathsf{UnMask})$ be a masking scheme with plaintext space $\mathcal{X}$ and masked value space $\mathcal{Y} = [0, |\mathcal{Y}|)$, and F be a PRF. If this triplet is admissible (and multiplication admissible) for some circuit class for which we have a semi-private HSS scheme with linear, offline/online sharing of semi-private inputs, then the construction of fig. 11 is an arithmetic garbling scheme for $\mathcal{G}$ wide-gate circuits. For a size s, depth D wide-gate circuit with n inputs and m outputs, the size of the garbled circuit and labels is $s \log(|\mathcal{Y}|) + m \log(|\mathcal{K}|) + (n + D + 1)\mathsf{poly}(\lambda)$.*

Proof.

- **Strong correctness.** Let us prove a stronger notion of correctness, namely that with all but negligible probability the following invariants are maintained for every wire w_i (whose depth is denoted d_i):

$$A_i: \qquad \text{“}y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)\text{”} \qquad (\forall i \geq 1)$$
$$B_{i,d'}: \text{“}w_{i,d'}^{(1)} - w_{i,d'}^{(0)} = \mathsf{sk}_{d'} \cdot \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)\text{”} \ (\forall i \geq 1, \forall d' \in [d_i, D])$$

The proof is completely analogous to that of theorem 28. For all $i \in [n]$, $\Pr[A_i] = 1$ by definition of $y_i \leftarrow \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i)$, and for all $d' \in [d_i, D]$ $\Pr[B_{i,d'}] = 1$ by definition of $\mathsf{Share}_{0,\mathsf{s-priv}}$ and $\mathsf{Share}_{1,\mathsf{s-priv}}$. Assume towards a contradiction that there is a non-negligible probability that there exists a wire index $i \geq n$ such that $(\neg A_i) \vee \bigvee_{d'=d_i}^{D}(\neg B_{i,d'})$. By a union bound, because there are only polynomially many wires, there exists a wire index $i \geq n$ such that $(\neg A_i) \vee \bigvee_{d'=d_i}^{D}(\neg B_{i,d'})$ with non-negligible probability. Let $i^\star$ be the smallest such index. It follows from its minimality that there exists a negligible function ε and a non-negligible function δ such that $\Pr\left[\bigwedge_{i=1}^{i^\star-1}\left(A_i \wedge \bigwedge_{d'=d_i}^{D} B_{i,d'}\right)\right] \leq \varepsilon$ and $\Pr\left[\neg A_{i^\star} \vee \bigvee_{d'=d_{i^\star}}^{D} \neg B_{i^\star,d'}\right] \geq \delta$. By essentially the same reduction to PRF security as used in the proof of strong correctness of theorem 28, because $\forall j \in [\ell], \tilde{w}_{i,d_i}^{(1)} - \tilde{w}_{i,d_i}^{(0)} = \mathsf{sk}_{d_i} \cdot \mathsf{Mask}_{F_{\vec{r}}(i_j)}(w_{i_j})$, it follows that $(y_{i^\star}^{(1)}, \tilde{w}_{i^\star,d_{i^\star}}^{(1)}) - (y_{i^\star}^{(0)}, \tilde{w}_{i^\star,d_{i^\star}}^{(0)}) = (1, \mathsf{sk}_{d_{i^\star}}) \cdot \mathsf{Mask}_{F_{\vec{r}}(i^\star)}(w_{i^\star})$ (and in turn $A_{i^\star} \wedge B_{i^\star,d_{i^\star}}$) with all but negligible probability. Similarly, by PRF security, for every $d' \in [d_{i^\star}, D-1]$, if $\bigwedge_{j \in [\ell]}(A_j \wedge B_{j,d'})$ then with all but negligible probability $A_{i^\star} \wedge B_{i^\star,d'+1}$. By combining all of these polynomially many hybrids, $(\neg A_i) \vee \bigvee_{d'=d_i}^{D}(\neg B_{i,d'})$ is negligible, which constitutes a contradiction.

- **Security.** We must show that the evaluator's view can be simulated using only the circuit and its output. We present a hybrid argument starting from the real garbled circuit and wire labels, with the goal of removing all dependence on the circuit inputs x_i from the view. The final hybrid will become the simulator.

$\mathcal{H}_1$. This is the real world. The evaluator sees the garbled circuit: k_{PRF}, $\mathsf{ek}_{1,d}$ for $d \in [0, D]$, $r_j^{(1)}$ for $j \in [\lambda]$, $s_d^{(1)}$ for $d \in [1, D]$, $\overline{y}_i^{(0)} \leftarrow (y_i^{(0)} \bmod |\mathcal{Y}|)$, and $R_j := F_{\vec{r}}(j)$ for each output gate index j. The evaluator also see its wire labels: $(y_i, w_i^{(1)}) \xleftarrow{\$} \mathsf{HSS.Share}_{1,\text{s-priv}}(1^\lambda, \mathsf{sk}_0, w_i^{(0)}, \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i))$.

$\mathcal{H}_2$. By strong correctness, with all but negligible probability every masked wire y_i satisfies computed by the evaluator satisfies $y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(w_i)$ for all i, where w_i is the plaintext value on the wire. Similarly to the proof of theorem 28, use strong correctness to rewrite each garbled gates $\overline{y}_i^{(0)}$ in terms of y_i and $y_i^{(1)}$:
$$y_i \leftarrow (y_i^{(1)} - \overline{y}_i^{(0)}) \bmod |\mathcal{Y}|$$
$$\Downarrow$$
$$\overline{y}_i^{(0)} \leftarrow (y_i^{(1)} - y_i) \bmod |\mathcal{Y}|$$

After this change, none of the garbler's HSS evaluation keys $\mathsf{ek}_{0,d}$ are used.

$\mathcal{H}_3$. Perform a sequence of hybrids $\mathcal{H}_3^0 \ldots, \mathcal{H}_3^{D+1}$, with $\mathcal{H}_3^0 = \mathcal{H}_2$ being the previous hybrid. In the change from $\mathcal{H}_3^d$ to $\mathcal{H}_3^{d+1}$, replace $\mathsf{ek}_{1,d}$ and all calls to $\mathsf{HSS.Share}_{\text{priv}}(1^\lambda, \mathsf{sk}_d, \ldots)$ or $\mathsf{HSS.Share}_{\text{s-priv}}(1^\lambda, \mathsf{sk}_d, \ldots)$ with their simulations from the semi-private security of HSS against party 1. These shares include $\mathsf{HSS.Share}_{\text{priv}}(1^\lambda, \mathsf{sk}_d, r_j)$ and $\mathsf{HSS.Share}_{\text{priv}}(1^\lambda, \mathsf{sk}_d, \mathsf{sk}_{d+1})$ (for $d < D$). For $d = 0$, these shares also include $\mathsf{HSS.Share}_{1,\text{s-priv}}(1^\lambda, \mathsf{sk}_0, w_i^{(0)}, \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i))$, and because these shares are semi-private their simulation will still be based on $y_i = \mathsf{Mask}_{F_{\vec{r}}(i)}(x_i)$.

We now argue that the change from $\mathcal{H}_3^d$ to $\mathcal{H}_3^{d+1}$ is indistinguishable. In $\mathcal{H}_3^d$, the key sk_d is only used to generate shares, since the share $s_d^{(1)}$ of sk_d under key sk_{d-1} was replaced with a simulation in the $\mathcal{H}_3^{d-1}$. Therefore, we can apply the semi-private security of HSS to sk_d, which takes us to $\mathcal{H}_3^{d+1}$.

$\mathcal{H}_4$. The last two changes are the same as in theorem 28: replace $F_{\vec{r}}$ by a random function, and then use one-time privacy of the masking scheme to replace non-output y_i with simulations. They work exactly as in theorem 28 After these changes, the distribution depends only on the output values of the garbled circuit, so it is a valid simulator. $\qquad\square$

5.3 Instantiating the Template and Applications to Sublinear-Size Garbling

The garbling schemes in section 5.2 are parameterized by a class of wide-gates and a masking scheme. In any instantiation, the wide-gate class and masking scheme must be admissible; that is, the function that unmasks, evaluates, then remasks must be in $\mathsf{RMS} \otimes \mathsf{VP}$.

Due to space constraints we only state the main theorems here, and defer the details to Appendix B.

Boolean Garbling For the Boolean case, putting Section together with theorems 18, 22, 28 and 31, we get rate 1 garbling for truth-table circuits.

Theorem 32. *Assuming either DCR or KDM-security of Damgård-Jurik, there exists a garbling scheme for boolean truth table wide-gate circuits. The size of the garbled circuit and labels is $s + m + (n + D + 1) \cdot \mathsf{poly}(\lambda)$ for DCR, or $s + m + (n + 1) \cdot \mathsf{poly}(\lambda)$ for KDM-security, if the truth table circuit has size s, depth D, n inputs, and m outputs.*

This result can also be applied to fan-in 2 layered boolean circuits. $\log \log(s)$-depth fan-in 2 circuits have at most $\log(s)$ inputs, and so can be represented in truth tables of size $\mathsf{poly}(s)$. Deep, but layered boolean circuits can then be chopped into chunks of depth $\log \log(s)$, and each bit of output of each chunk can be represented as a truth table. Therefore, m-output layered boolean circuits with size s can equivalently be written as fan-in $\log \log(s)$ boolean truth table circuits of size $n + m + s/\log \log(s)$ [Cou19].

Theorem 33. *Assuming either DCR or KDM-security of Damgård-Jurik, there exists a sub-linear communication garbling scheme for layered boolean circuits. The size of the garbled circuit and labels is $O(s/\log \log(s) + m) + (n + D + 1) \cdot \mathsf{poly}(\lambda)$ for DCR, or $O(s/\log \log(s) + m) + (n + 1) \cdot \mathsf{poly}(\lambda)$ for KDM-security, if the boolean circuit has size s, depth D, n inputs, and m outputs.*

Arithmetic Garbling. For the arithmetic case, putting together Section with theorems 18, 22, 28 and 31, we get rate 1 arithmetic garbling for B-bounded arithmetic circuits with multivariate polynomial gates, if $\log(B) = \omega(B)$.

Theorem 34. *Assuming either DCR or KDM-security of Damgård-Jurik, there exists a gar-bling scheme for B-bounded arithmetic circuits with multivariate polynomial wide-gates of logarithmic degree and polynomial number of terms. The size of the garbled circuit and labels is $(s + m)(\lambda + \log B + 2) + (n + D + 1) \cdot \mathsf{poly}(\lambda, \log B)$ for DCR, or $(s + m)(\lambda + \log B + 2) + (n + 1) \cdot \mathsf{poly}(\lambda, \log B)$ for KDM-security, if the wide-gate circuit has size s, depth D, n inputs, and m outputs.*

And again, we can use a standard technique to convert to sublinear communication garbling of fan-in 2 layered arithmetic circuits.

Theorem 35. *Assuming either DCR or KDM-security of Damgård-Jurik, there exists a sub-linear communication garbling scheme for B-bounded layered arithmetic circuits. The size of the garbled circuit and labels is $O((s/\log \log(s) + m)(\lambda + \log B)) + (n + D + 1) \cdot \mathsf{poly}(\lambda, \log B)$ for DCR, or $O((s/\log \log(s) + m)(\lambda + \log B)) + (n + 1) \cdot \mathsf{poly}(\lambda, \log B)$ for KDM-security, if the arithmetic circuit has size s, depth D, n inputs, and m outputs.*

Proof. By [Cou19], any fan-in 2 layered arithmetic circuit can be divided into layers of depth $\log \log(s)$, giving an equivalent wide-gate circuit of size $O(s/\log \log(s) + n + m)$. Each gate of this circuit evaluates a multivariate polynomial of degree at most $2^{\log \log(s)} = \log(s)$ and at most $2^{\log \log(s)} = \log(s)$ variables. By stars-and-bars counting, this polynomial has at most $\binom{2 \log(s)}{\log(s)} \le s^2$ monomials. Therefore, gates of this circuit can be evaluated by the garbling scheme of theorem 34. $\qquad\square$

Acknowledgments

References

AIK11.   Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.

Bar89.   David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

BCM23.   Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In Carmit Hazay and Martijn Stam, editors, *EURO-CRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 159–189. Springer, Cham, April 2023.

BGG+14.   Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014.

BGI16.   Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Berlin, Heidelberg, August 2016.

BLLL23.   Marshall Ball, Hanjun Li, Huijia Lin, and Tianren Liu. New ways to garble arithmetic circuits. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 3–34. Springer, Cham, April 2023.

BMO+25.   Amik Raj Behera, Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Privately constrained PRFs from DCR: Puncturing and bounded waring rank. 2025. Cryptology ePrint Archive.

Cle90.   Richard Cleve. Towards optimal simulations of formulas by bounded-width programs. In *22nd ACM STOC*, pages 271–277. ACM Press, May 1990.

CMPR23.   Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Constrained pseudo-random functions from homomorphic secret sharing. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 194–224. Springer, Cham, April 2023.

Cou19.   Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 473–503. Springer, Cham, May 2019.

DIO21.   Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

DJ01.   Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Berlin, Heidelberg, February 2001.

FNO15.   Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 191–219. Springer, Berlin, Heidelberg, April 2015.

GKP+13.   Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

HK20.   David Heath and Vladimir Kolesnikov. Stacked garbling - garbled circuit proportional to longest execution path. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 763–792. Springer, Cham, August 2020.

HK21a.   David Heath and Vladimir Kolesnikov. One hot garbling. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 574–593. ACM Press, November 2021.

HK21b.   David Heath and Vladimir Kolesnikov. LogStack: Stacked garbling with $O(b \log b)$ computation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 3–32. Springer, Cham, October 2021.

HKN24.   David Heath, Vladimir Kolesnikov, and Lucien K. L. Ng. Garbled circuit lookup tables with logarithmic number of ciphertexts. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 185–215. Springer, Cham, May 2024.

HLL23.   Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023.

HO12.   Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 627–643. Springer, Berlin, Heidelberg, May 2012.

ILL24.   Yuval Ishai, Hanjun Li, and Huijia Lin. Succinct partial garbling from groups and applications. Cryptology ePrint Archive, Paper 2024/2073, 2024.

IW14.   Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Berlin, Heidelberg, July 2014.

LWYY24. Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. Garbled circuits with 1 bit per gate. Cryptology ePrint Archive, Paper 2024/1988, 2024.

MORS24. Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. Rate-1 arithmetic garbling from homomorphic secret sharing. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part IV*, volume 15367 of *LNCS*, pages 71–97. Springer, Cham, December 2024.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Cham, October 2021.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Berlin, Heidelberg, May 1999.

RR21. Mike Rosulek and Lawrence Roy. Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 94–124, Virtual Event, August 2021. Springer, Cham.

RS21. Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 687–717, Virtual Event, August 2021. Springer, Cham.

Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

A Offline-Online HSS Proofs

A.1 From KDM-security of Damgård-Jurik

In this section, we prove theorem 18, to show that fig. 6 is a Semi-Private HSS scheme (definition 16) with linear offline/online sharing of semi-private inputs and supporting authenticated evaluation (definition 17). We prove each property of the semi-private HSS scheme separately.

A.1.1 Correctness and authenticated correctness. Let $\lambda \in \mathbb{N}$ be a security parameter, $B \leq 2^{(\zeta-1-\varepsilon)\cdot\lambda_{\mathsf{DCR}}/2}$ a bound, let C be an $n_{\mathsf{s\text{-}priv}}$-input circuit over $\mathbb{Z}$, let C_{rm} be a restricted-multiplication n_{priv}-input circuit over $\mathbb{Z}$, let $(x_{\mathsf{s\text{-}priv},i})_{i\in[n_{\mathsf{s\text{-}priv}}]}$ and $(x_{\mathsf{priv},i})_{i\in[n_{\mathsf{priv}}]}$ be B-admissible inputs for C and C_{rm} respectively.

Let us prove that correctness (definition 16) holds with probability at least $p\cdot(1-N^{-\varepsilon})-\varepsilon_{\mathsf{PRF}}$ and that authenticated correctness (definition 17) holds with probability at least $p-\varepsilon_{\mathsf{PRF}}$, where

$$p := 1 - (n_{\mathsf{s\text{-}priv}} + 2n_{\mathsf{priv}} + |C_\times| + |C'_\times|) \cdot N^{-\varepsilon}$$

(with $|C_\times|$ and $|C'_\times|$ denoting the number of multiplication gates in C and C_{rm} respectively) and $\varepsilon_{\mathsf{PRF}}$ is tied to the security of PRF.

The proof boils down to showing that all of the comments in the pseudocode of fig. 6 are invariants holding with all but negligible probability.

Idealised PRF. For simplicity, we first consider an idealised version of the HSS scheme of fig. 6 where PRF is replaced with a truly random function (*i.e.* the evaluation keys contain the same polynomial-size random string—because PRF has a polynomial-sized domain—instead of the short PRF key). We will later remove this restriction by a reduction to the security of PRF.

Consider the following random variables: (where $\mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,0}$ and $\mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,1}$ are random variables implicitly defined by Eval)

1. $(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$;
2. $(x_{\mathsf{priv},i}^{(0)}, x_{\mathsf{priv},i}^{(1)}) \xleftarrow{\$} \mathsf{Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x_{\mathsf{priv},i})$
3. $y_{\mathsf{s\text{-}priv},i}^{(0)} \xleftarrow{\$} \mathsf{Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk})$
4. $y_{\mathsf{s\text{-}priv},i}^{(1)} \xleftarrow{\$} \mathsf{Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y_{\mathsf{s\text{-}priv},i}, \mathsf{st})$
5. $z_0 \xleftarrow{\$} \mathsf{Eval}(0, \mathsf{ek}_0, (x_{\mathsf{priv},i}^{(0)})_{i=1}^{n_{\mathsf{priv}}}, (y_{\mathsf{s\text{-}priv},i}^{(0)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}, (C$
6. $z_1 \xleftarrow{\$} \mathsf{Eval}(1, \mathsf{ek}_1, (x_{\mathsf{priv},i}^{(1)})_{i=1}^{n_{\mathsf{priv}}}, (y_{\mathsf{s\text{-}priv},i}^{(1)})_{i=1}^{n_{\mathsf{s\text{-}priv}}}, (C, C_{\mathsf{rm}}))$
7. $\tilde{z}_0 \leftarrow (z_0, \mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,0})$
8. $\tilde{z}_1 \leftarrow (z_1, \mathsf{sh}_{|C|+|C_{\mathsf{rm}}|,1})$
9. $z \xleftarrow{\$} z_1 - z_0$
10. $\tilde{z} \xleftarrow{\$} \tilde{z}_1 - \tilde{z}_0$

Observe that if we show that $\Pr[z = C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot C_{\mathsf{rm}}(\vec{x}_{\mathsf{priv}})] \geq 1 - \mathsf{negl}(\lambda)$ and $\Pr[\tilde{z} = (1, \varphi) \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot C_{\mathsf{rm}}(\vec{x}_{\mathsf{priv}})] \geq 1 - \mathsf{negl}(\lambda)$ then we will have proven both correctness and authenticated correctness.

Consider the following events:

$$E_{\text{corr}}: \qquad \text{``}\mathsf{Dec}_\varphi(c_{\text{inv}}) \equiv \varphi^{-1} \bmod N^\zeta\text{''}$$

$$E_{\text{corr},j}: \qquad \text{``}\mathsf{Dec}_\varphi(c_j) \equiv x_{\text{priv},j} \bmod N^\zeta\text{''} \qquad\qquad (\text{for } j \in [n_{\text{priv}}])$$

$$E_0: \qquad \text{``}\mathsf{sh}_{0,1} - \mathsf{sh}_{0,0} = \varphi \cdot 1\text{''}$$

$$E_i: \qquad \text{``}\mathsf{sh}_{i,1} - \mathsf{sh}_{i,0} = \varphi \cdot y_{\text{s-priv},i}\text{''} \qquad\qquad (\text{for } i \in [n_{\text{s-priv}}])$$

$$E_i: \qquad \text{``}\mathsf{sh}_{i,1} - \mathsf{sh}_{i,0} = \varphi \cdot P_i(\vec{y}_{\text{s-priv}})\text{''} \qquad (\text{for } i \in [n_{\text{s-priv}}+1, |C|])$$

$$E_i': \quad \text{``}\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} = \varphi \cdot C(\vec{y}_{\text{s-priv}}) \cdot Q_i(\vec{x}_{\text{priv}})\text{''} \qquad (\text{for } i \in [|C_{\text{rm}}|])$$

where for $i \in [n_{\text{s-priv}}+1, |C|]$, P_i is the polynomial computed by the ith gate of C, while for $i \in [|C_{\text{rm}}|]$, Q_i is the polynomial computed by the ith gate of C_{rm}.

Lemma 36. $\Pr[E_0] = 1$.

Lemma 36 follows immediately from the definition of $\mathsf{sh}_{0,1}$ and $\mathsf{sh}_{0,0}$.

Lemma 37. $\forall i \in [n_{\text{s-priv}}], \Pr[E_i] \geq 1 - N^{-\varepsilon}$.

> *Proof of lemma Lemma 37.* By definition of the $\mathsf{sh}_{i,1}$ and $\mathsf{sh}_{i,0}$ ($i \in [n_{\text{s-priv}}]$), the following relation holds with probability 1:
>
> $$\forall i \in [n_{\text{s-priv}}], \mathsf{sh}_{i,1} - \mathsf{sh}_{i,0} \equiv \varphi \cdot x_{\text{s-priv},i} \bmod N^\zeta$$
>
> Because the offline and the online inputs are all B-admissible, and because $\varphi \cdot B \leq N \cdot B = N^{\zeta-\varepsilon} << N^\zeta$, we get the desired results by invoking Lemma 9 (in particular because the $\mathsf{sh}_{i,0}$ for $i \in [n_{\text{s-priv}}]$ are sampled uniformly at random from $\mathbb{Z}/N^\zeta\mathbb{Z}$). $\qquad\square$

Lemma 38. Let $i \in [n_{\text{s-priv}}+1, |C|]$. If g_i (the ith gate of C) is the constant gate, then $\Pr[E_i] = 1$.

> *Proof of Lemma 38.* If g_i is the constant gate 1 then by definition $E_i \equiv E_0$ so $\Pr[E_i] = 1$ by Lemma 36. $\qquad\square$

Lemma 39. Let $i \in [n_{\text{s-priv}}+1, |C|]$. If g_i (the ith gate of C) is the jth input gate, then $\Pr[E_i|E_j] = 1$.

> *Proof of Lemma 39.* If g_i is the jth input then by definition $E_i \equiv E_j$. By Lemma 37, $\Pr[E_j] \neq 0$ so $\Pr[E_i|E_j] = 1$ (because any non-impossible event is certain when conditioned on itself). $\qquad\square$

Lemma 40. Let $i \in [n_{\text{s-priv}}+1, |C|]$. If g_i (the ith gate of C) is an addition whose parents are g_{i_L} and g_{i_R}, then $\Pr[E_i|E_{i_L}, E_{i_R}] = 1$.

> *Proof of Lemma 40.* If g_i is an addition gate, whose parents we denote i_R and i_L, then,

conditioned on E_{i_L} and E_{i_R}:

$$\begin{aligned}
\mathsf{sh}_{i,1} - \mathsf{sh}_{i,0} &= (\mathsf{sh}_{i_R,1} + \mathsf{sh}_{i_L,1}) - (\mathsf{sh}_{i_R,0} + \mathsf{sh}_{i_L,0}) \\
&= (\mathsf{sh}_{i_L,1} - \mathsf{sh}_{i_L,0}) + (\mathsf{sh}_{i_R,1} - \mathsf{sh}_{i_R,0}) \\
&= \varphi \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}}) + \varphi \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}}) \\
&= \varphi \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}})
\end{aligned}$$

Therefore $\Pr[E_i | E_{i_L}, E_{i_R}] = 1$. $\qquad\square$

Lemma 41. *Let $i \in [n_{\mathsf{s\text{-}priv}} + 1, |C|]$. If g_i (the ith gate of C) is a scalar multiplication gate whose parent is g_j, then $\Pr[E_i | E_j] = 1$.*

Proof of Lemma 41. If g_i is the scalar multiplication $\cdot_\alpha$ with predecessor g_j, then, conditioned on E_j:

$$\begin{aligned}
\mathsf{sh}_{i,1} - \mathsf{sh}_{i,0} &= \alpha \cdot \mathsf{sh}_{j,1} - \alpha \cdot \mathsf{sh}_{j,0} \\
&= \alpha \cdot (\mathsf{sh}_{j,1} - \mathsf{sh}_{j,0}) \\
&= \alpha \cdot \varphi \cdot P_j(\vec{y}_{\mathsf{s\text{-}priv}}) \\
&= \varphi \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}})
\end{aligned}$$

Therefore $\Pr[E_i | E_j] = 1$. $\qquad\square$

Lemma 42. *Let $i \in [n_{\mathsf{s\text{-}priv}} + 1, |C|]$. If g_i (the ith gate of C) is a multiplication whose parents are g_{i_L} and g_{i_R}, then $\Pr[E_i | E_{i_L}, E_{i_R}, E_{\mathsf{corr}}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 42. Let $i \in [n_{\mathsf{s\text{-}priv}} + 1, |C|]$ such that g_i is a multiplication gate whose two operands are g_{i_L} and g_{i_R} (with $i_L, i_R < i$). Denote P_{i_L} and P_{i_R} the polynomials computed by g_{i_L} and g_{i_R} respectively. By core lemma 1, conditioned on E_{i_L}, E_{i_R}, and E_{corr}:

$$\begin{aligned}
\left(-\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\mathsf{sh}_{i_L,1} \cdot \mathsf{sh}_{i_R,1}} \right) + P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot \mathsf{sh}_{i_R,1} + P_{i_R}(\vec{x}_{\mathsf{priv}}) \cdot \mathsf{sh}_{i_L,1} \right) \\
- \left(-\mathsf{DDLog}\left(c_{\mathsf{inv}}^{\mathsf{sh}_{i_L,0} \cdot \mathsf{sh}_{i_R,0}} \right) \right) \equiv \varphi \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}}) \bmod N^\zeta
\end{aligned}$$

We conclude, by Lemma 9 (because $\varphi \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}}) \leq N \cdot B \leq N^{\zeta - \varepsilon}$), that $\Pr\left[E_i | E_{i_L}, E_{i_R}, E_{\mathsf{corr}} \right] \geq 1 - N^{-\varepsilon}$. $\qquad\square$

Lemma 43. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is a constant gate then $\Pr[E'_i | E_{|C|}] = 1$.*

Proof of Lemma 43. If g'_i is the constant gate 1, then by definition $E'_i \equiv E_{|C|}$ so $\Pr[E'_i | E_{|C|}] = 1$. $\qquad\square$

Lemma 44. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is an addition whose parents are g'_{i_L} and g'_{i_R}, then $\Pr[E'_i | E'_{i_L}, E'_{i_R}] = 1$.*

Proof of Lemma 44. If g'_i is an addition gate, whose parents are indexed i_R and i_L,

then, conditioned on E'_{i_L} and E'_{i_R}:

$$\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} = (\mathsf{sh}_{|C|+i_R,1} + \mathsf{sh}_{|C|+i_L,1}) - (\mathsf{sh}_{|C|+i_R,0} + \mathsf{sh}_{|C|+i_L,0})$$
$$= (\mathsf{sh}_{|C|+i_L,1} - \mathsf{sh}_{|C|+i_L,0}) + (\mathsf{sh}_{|C|+i_R,1} - \mathsf{sh}_{|C|+i_R,0})$$
$$= \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_{|C|+i_L}(\vec{x}_{\mathsf{priv}}) + \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_{|C|+i_R}(\vec{x}_{\mathsf{priv}})$$
$$= \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_i(\vec{x}_{\mathsf{priv}})$$

Therefore $\Pr[E'_i | E'_{i_L}, E'_{i_R}] = 1$. $\qquad\square$

Lemma 45. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is a scalar multiplication gate whose parent is g'_j, then $\Pr[E'_i | E'_j] = 1$.*

Proof of Lemma 45. If g'_i is the scalar multiplication $\cdot_\alpha$ with predecessor g'_j, then, conditioned on E'_j:

$$\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} = \alpha \cdot \mathsf{sh}_{|C|+j,1} - \alpha \cdot \mathsf{sh}_{|C|+j,0}$$
$$= \alpha \cdot (\mathsf{sh}_{|C|+j,1} - \mathsf{sh}_{|C|+j,0})$$
$$= \alpha \cdot \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_j(\vec{x}_{\mathsf{priv}})$$
$$= \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_i(\vec{x}_{\mathsf{priv}})$$

Therefore $\Pr[E'_i | E'_j] = 1$. $\qquad\square$

Lemma 46. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is the jth input gate, then $\Pr[E'_i | E_{|C|}, E_{\mathsf{priv},j}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 46. If g'_i is the jth input gate $x_{\mathsf{priv},j}$ and we condition on $E_{|C|}$, then by Lemma 7

$$\mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|,1}}\right) - \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|,0}}\right) \equiv \mathsf{DJ.Dec}_{N,\zeta,\varphi}(c_j) \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot \varphi \mod N^\zeta$$

If we further condition on $E_{\mathsf{priv},j}$, then

$$\mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|,1}}\right) - \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|,0}}\right) \equiv \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j} \mod N^\zeta$$

Finally, because $C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j} \leq N \cdot B^2 \leq N^{\zeta-\varepsilon}$, (still conditioned on $E_{|C|}$ and $E_{\mathsf{priv},j}$) E'_i holds with probability at least $1 - N^{-\varepsilon}$ by Lemma 9. $\qquad\square$

Lemma 47. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is a multiplication gate whose parents are the jth input gate and g_{i_R}, then $\Pr[E'_i | E'_{i_R}, E_{\mathsf{corr},j}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 47. If g'_i is the multiplication of the jth input gate $x_{\mathsf{priv},j}$ and gate g'_{i_R}, then, conditioning on E'_{i_R}, by Lemma 7

$$\mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|+i_R,1}}\right) - \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|+i_R,0}}\right)$$
$$\equiv \mathsf{DJ.Dec}_{N,\zeta,\varphi}(c_j) \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_{i_R}(\vec{x}_{\mathsf{priv}}) \cdot \varphi \mod N^\zeta$$

If we further condition on $E_{\mathsf{corr},j}$, then

$$\mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|+i_R,1}}\right) - \mathsf{DDLog}\left((c_j)^{\mathsf{sh}_{|C|+i_R,0}}\right) \equiv \varphi \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_i(\vec{x}_{\mathsf{priv}}) \mod N^\zeta$$

Finally, because $C(\vec{y}_{\text{s-priv}}) \cdot Q_i(\vec{x}_{\text{priv}}) \le N \cdot B^2 \le N^{\zeta - \varepsilon}$, (still conditioned on $E_{|C|+i_R}$ and $E_{\text{priv},j}$) E_i' holds with probability at least $1 - N^{-\varepsilon}$ by Lemma 9. $\qquad\square$

Lemma 48. *Authenticated correctness with an idealised PRF (i.e. the event "$\tilde{y} = \varphi \cdot C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})$") holds with probability at least $1 - (n_{\text{s-priv}} + n_{\text{priv}} + |C_\times| + |C_\times'|) \cdot N^{-\varepsilon}$, where $|C_\times|$ and $|C_\times'|$ are the number of multiplication gates in C and C_{rm} respectively.*

Proof of Lemma 48. To simplify notations, we rename/order the events $E_{\text{corr}}, (E_{\text{corr},j})_{j \in [n_{\text{priv}}]}, (E_i)_{i \in [0,|C|]}, (E_i')_{i \in [|C_{\text{rm}}|]}$ as $A_1, \ldots, A_{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2}$:

1. $A_1 := E_{\text{corr}}$
2. For $j \in [n_{\text{priv}}]$, $A_{j+1} := E_{\text{corr},j}$
3. For $j \in [0, |C|]$, $A_{n_{\text{s-priv}}+2+j} := E_j$
4. For $j \in [|C_{\text{rm}}|]$, $A_{n_{\text{s-priv}}+2+|C|+j} := E_j'$

Observe that

$$
\begin{aligned}
\Pr[E_{|C_{\text{rm}}|}'] &= \Pr[A_{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2}] \\
&\ge \Pr[\bigcap_{i=1}^{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2} A_i] \\
&= 1 - \Pr[\bigcup_{i=1}^{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2} \overline{A_i}] \\
&= 1 - \Pr[\bigsqcup_{i=1}^{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2} (\overline{A_i} \cap \bigcap_{1 \le j < i} A_j)] \\
&= 1 - \sum_{i=1}^{n_{\text{priv}}+|C|+|C_{\text{rm}}|+2} \Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j]
\end{aligned}
$$

Let us now upper bound each of the $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j]$, which in turn allows us to lower bound $\Pr[E_{|C_{\text{rm}}|}']$.

1. $\Pr[\overline{A_1}] = \Pr[\overline{E_{\text{corr}}}] = 0$ by perfect correct of the Damgård-Jurik encryption scheme.
2. For $i \in [2, n_{\text{priv}}+1]$, $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i}] = \Pr[\overline{E_{\text{corr},i-1}}] = 0$ by perfect correct of the Damgård-Jurik encryption scheme.
3. $\Pr[\overline{A_{n_{\text{priv}}+2}} \cap \bigcap_{1 \le j < n_{\text{priv}}+2} A_j] \le \Pr[\overline{A_{n_{\text{priv}}+2}}] = \Pr[\overline{E_0}] = 0$ by Lemma 36.
4. For $i \in [n_{\text{priv}}+3, n_{\text{priv}}+n_{\text{s-priv}}+2]$, $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i}] = \Pr[\overline{E_{j-n_{\text{priv}}-2}}] \le N^{-\varepsilon}$ by Lemma 37.
5. For $i \in [n_{\text{priv}} + n_{\text{s-priv}} + 3, n_{\text{priv}} + n_{\text{s-priv}} + |C| + 2]$,
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}$ is the constant gate, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}] = 0$ by Lemma 38.
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}$ is the jth input gate, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le k < i} A_k] \le \Pr[\overline{A_i} \cap A_{n_{\text{priv}}+j+2}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}} \cap E_j] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_j] \cdot \Pr[E_{\text{s-priv},j}] = 0$ by Lemma 39.
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}$ is an addition gate whose parents are g_{i_L} and g_{i_R}, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i} \cap A_{n_{\text{s-priv}}+2+i_L} \cap A_{n_{\text{s-priv}}+2+i_R}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}} \cap E_{i_L} \cap E_{i_R}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_{i_L}, E_{i_R}] \cdot \Pr[E_{i_L} \cap E_{i_R}] \le \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_{i_L}, E_{i_R}] = 0$ by Lemma 40.
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}$ is a scalar multiplication gate whose parent is g_j, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le k < i} A_k] \le \Pr[\overline{A_i} \cap A_{n_{\text{s-priv}}+2+j}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}} \cap E_j] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_j] \cdot \Pr[E_j] \le \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_j] = 0$ by Lemma 41.
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}$ is a multiplication gate whose parents are g_{i_L} and g_{i_R}, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i} \cap A_{n_{\text{priv}}+2+i_L} \cap A_{n_{\text{priv}}+2+i_R} \cap A_1] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}} \cap E_{i_L} \cap E_{i_R} \cap E_{\text{corr}}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_{i_L}, E_{i_R}, E_{\text{corr}}] \cdot \Pr[E_{i_L} \cap E_{i_R} \cap E_{\text{corr}}] \le \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-2}}|E_{i_L}, E_{i_R}, E_{\text{corr}}] \le N^{-\varepsilon}$ by Lemma 42.
6. For $i \in [n_{\text{s-priv}} + n_{\text{priv}} + |C| + 3, n_{\text{s-priv}} + n_{\text{priv}} + |C| + |C_{\text{rm}}| + 2]$,
   - If $g_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}'$ is the constant gate, then $\Pr[\overline{A_i} \cap \bigcap_{1 \le j < i} A_j] \le \Pr[\overline{A_i}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}'}] = 0$ by Lemma 43.

41

- If $g'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}$ is an addition gate whose parents are g'_{i_L} and g'_{i_R}, then $\Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j] \leq \Pr[\overline{A_i} \cap A_{n_{\text{s-priv}}+n_{\text{priv}}+|C|+2+i_L} \cap A_{n_{\text{s-priv}}+n_{\text{priv}}+|C|+2+i_R}] = \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}} \cap E'_{i_L} \cap E'_{i_R}] = \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}}|E'_{i_L}, E'_{i_R}] \cdot \Pr[E'_{i_L} \cap E'_{i_R}] \leq \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}}|E'_{i_L}, E'_{i_R}] = 0$ by Lemma 44.

- If $g'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}$ is an scalar multiplication gate whose parent is g'_j, then $\Pr[\overline{A_i} \cap \bigcap_{1 \leq k < i} A_k] \leq \Pr[\overline{A_i} \cap A_j] = \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}} \cap E'_j] = \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}}|E'_j] \cdot \Pr[E'_j] \leq \Pr[\overline{E'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}}|E'_j] = 0$ by Lemma 45.

- If $g'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}$ is the jth input gate, then $\Pr[\overline{A_i} \cap \bigcap_{1 \leq k < i} A_k] \leq \Pr[\overline{A_i} \cap A_{n_{\text{s-priv}}+n_{\text{priv}}+|C|+2}, A_{j+1}, A_1] = \Pr[\overline{E'_i} \cap E_{|C|} \cap E_{\text{corr},j} \cap E_{\text{corr}}] = \Pr[\overline{E'_i}|E_{|C|}, E_{\text{corr},j}, E_{\text{corr}}] \cdot \Pr[E_{|C|} \cap E_{\text{corr},j} \cap E_{\text{corr}}] \leq \Pr[\overline{E'_i}|E_{|C|}, E_{\text{corr},j}, E_{\text{corr}}] \leq N^{-\varepsilon}$ by Lemma 46.

- If $g'_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}$ is a multiplication gate whose parents are the jth input and g'_{i_R}, then $\Pr[\overline{A_i} \cap \bigcap_{1 \leq k < i} A_k] \leq \Pr[\overline{A_i} \cap A_{j+1} \cap A_{n_{\text{s-priv}}+n_{\text{priv}}+i_R+2} \cap A_1] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}'} \cap E_{\text{corr},j} \cap E'_{i_R} \cap E_{\text{corr}}] = \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}'}|E_{\text{corr},j}, E'_{i_R}, E_{\text{corr}}] \cdot \Pr[E_{\text{corr},j} \cap E'_{i_R} \cap E_{\text{corr}}] \leq \Pr[\overline{E_{i-n_{\text{s-priv}}-n_{\text{priv}}-|C|-2}'}|E_{\text{corr},j}, E'_{i_R}, E_{\text{corr}}] \leq N^{-\varepsilon}$ by Lemma 47.

By combining all of the above, $\Pr[E'_{|C_{\text{rm}}|}] \geq 1 - (n_{\text{s-priv}} + n_{\text{priv}} + |C_\times| + |C'_\times|) \cdot N^{-\varepsilon}$.[a] Because $E'_{|C_{\text{rm}}|}$ coincides with authenticated correctness, this concludes the proof. $\qquad\square$

---

[a] This bound is obtained by making the assumption that there are no duplicate input gates, *i.e.* that there are at most $n_{\text{s-priv}}$ input gates in C and at most n_{priv} in C_{rm}.

Lemma 49. *Correctness with an idealised PRF (i.e. the event "$y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})$") holds with probability at least $(1 - N^{-\varepsilon}) \cdot (1 - (n_{\text{s-priv}} + n_{\text{priv}} + |C_\times| + |C'_\times|) \cdot N^{-\varepsilon})$, where $|C_\times|$ and $|C'_\times|$ are the number of multiplication gates in C and C_{rm} respectively.*

Proof of Lemma 49. By Lemma 9, $\Pr["y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})"|E'_{|C_{\text{rm}}|}, E_{\text{corr}}] \geq 1 - N^{-\varepsilon}$ (using the fact that $C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}}) \leq B^2 \leq N^{\zeta - \varepsilon}$).

$$\Pr["y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})"] \geq \Pr["y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})", E'_{|C_{\text{rm}}|}, E_{\text{corr}}]$$
$$= \Pr["y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})"|E'_{|C_{\text{rm}}|}, E_{\text{corr}}] \cdot \Pr[E'_{|C_{\text{rm}}|}, E_{\text{corr}}]$$

Because $\Pr[E_{\text{corr}}] = 0$ by perfect correctness of the Damgård-Jurik encryption scheme, and $\Pr[E'_{|C_{\text{rm}}|}] \geq 1 - (n_{\text{s-priv}} + n_{\text{priv}} + |C_\times| + |C'_\times|) \cdot N^{-\varepsilon}$ by Lemma 48, $\Pr["y = C(\vec{y}_{\text{s-priv}}) \cdot C_{\text{rm}}(\vec{x}_{\text{priv}})"] \geq (1 - N^{-\varepsilon}) \cdot 1 - (n_{\text{s-priv}} + n_{\text{priv}} + |C_\times| + |C'_\times|) \cdot N^{-\varepsilon}$, which concludes the proof. $\qquad\square$

Real PRF. We now no longer assume that PRF is instantiated as a truly random function. Consider the adversary which, tasked with distinguishing outputs of PRF and a random oracle, runs the "HSS correctness experiment" and outputs 1 if and only if correctness holds. By security of PRF, the advantage of this adversary can be at most ε_{PRF}, and since we have already established that correctness (respectively authenticated correctness) in the ideal world holds with probability at least $p \cdot (1 - N^{-\varepsilon})$ (respectively p), where

$$p := \left[(1 - N^{-\varepsilon})^{|C_\times|} \cdot (1 - n_{\text{s-priv}} \cdot N^{-\varepsilon} - \varepsilon_{\text{corr}}) - n_{\text{priv}} \cdot N^{-\varepsilon}\right] \cdot (1 - N^{-\varepsilon})^{|C'_\times|+n_{\text{priv}}},$$

it follows that correctness (respectively authenticated correctness) in the real world holds with probability at least $p \cdot (1 - N^{-\varepsilon}) - \varepsilon_{\mathsf{PRF}}$ (respectively $p - \varepsilon_{\mathsf{PRF}}$).

A.1.2 Security against party $\sigma = 0$. We need to simulate the view of party $\sigma = 0$ using just the numbers $n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}}$. We present a hybrid argument, in which the last world the view of party 0 will depend only on these values. This last hybrid world becomes the simulation.

$\mathcal{H}_1$. This is the real world. The adversary sees $\mathsf{ek}_0 \leftarrow (N, c_{\mathsf{inv}}, k_{\mathsf{PRF}}, \mathsf{sh}_{1,0})$, $x_i^{(0)}$ (output by $\mathsf{Share}_{\mathsf{priv}}$) for $i \in [n_{\mathsf{priv}}]$, and $y_i^{(0)}$ (output by $\mathsf{Share}_{1,\mathsf{s\text{-}priv}}$) for $i \in n_{\mathsf{s\text{-}priv}}$. Notice that the only thing in the adversary's view that depends on private information that the simulator will not know are the Damgård-Jurik ciphertexts $x_i^{(0)}$.

$\mathcal{H}_2$. Notice that the only part of the adversary's view that φ is used for is $c_{\mathsf{inv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(\varphi^{-1} \bmod N^\zeta)$. Replace this with $c_{\mathsf{inv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(0)$. This is indistinguishable assuming KDM security of Damgård-Jurik.

$\mathcal{H}_3$. Now that the secret key φ is unused, we can simulate the encryptions $x_i^{(0)} \xleftarrow{\$} \mathsf{DJ.Enc}_N(x)$ as encryptions of zero $x_i^{(0)} \xleftarrow{\$} \mathsf{DJ.Enc}_N(x)$. This is indistinguishable by CPA security of Damgård-Jurik (which is implied by KDM security). The adversary's view is now being simulated using only the number of inputs, n_{priv} and $n_{\mathsf{s\text{-}priv}}$.

A.1.3 Security against party $\sigma = 1$. We need to simulate the view of party $\sigma = 1$ using just the numbers $n_{\mathsf{priv}}, n_{\mathsf{s\text{-}priv}}$ of shares, and the values of the semi-private shares y_i. We present a hybrid argument, in which the last world the view of party 1 will depend only on these values. This last hybrid world becomes the simulation.

$\mathcal{H}_1$. This is the real world. The adversary sees $\mathsf{ek}_1 = (N, c_{\mathsf{inv}}, k_{\mathsf{PRF}}, \mathsf{sh}_{1,1})$, $x_i^{(1)}$ (output by $\mathsf{Share}_{\mathsf{priv}}$) for $i \in [n_{\mathsf{priv}}]$, and $y_i^{(1)} = (y_i, \mathsf{sh}_{i,1})$ (output by $\mathsf{Share}_{1,\mathsf{s\text{-}priv}}$) for $i \in n_{\mathsf{s\text{-}priv}}$.

$\mathcal{H}_2$. Instead of sampling $\mathsf{sh}_{i,0} \xleftarrow{\$} \mathbb{Z}/N^\zeta\mathbb{Z}$ and setting $\mathsf{sh}_{i,1} \leftarrow (\varphi \cdot y_i + \mathsf{sh}_{i,0}) \bmod N^\zeta$, sample $\mathsf{sh}_{i,1} \xleftarrow{\$} \mathbb{Z}/N^\zeta\mathbb{Z}$. This is an identical distribution because $\mathsf{sh}_{i,0}$ is unused.

$\mathcal{H}_3$. Make a similar change to $\mathsf{sh}_{1,1}$. Instead of sampling $\mathsf{sh}_{1,0} \xleftarrow{\$} [0, N]$ and setting $\mathsf{sh}_{1,1} \leftarrow \varphi + \mathsf{sh}_{1,0}$, sample $\mathsf{sh}_{1,1} \xleftarrow{\$} [N, 2N]$ (and $\mathsf{sh}_{1,1}$ is unused). This shifts the distribution of $\mathsf{sh}_{1,1}$ from uniform on $[\varphi, N + \varphi]$ to uniform on $[N, 2N]$. Therefore, this change has statistical distance of
$$\frac{N - \varphi}{N + 1} = \frac{N - (pq - p - q + 1)}{N + 1} = \frac{p + q - 1}{N + 1} = O(N^{-\frac{1}{2}}).$$
where p and q are the factors of the semi-prime N, each of size roughly $N^{\frac{1}{2}}$.

$\mathcal{H}_4$. Notice that φ is now used in only one place: $c_{\mathsf{inv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(\varphi^{-1} \bmod N^\zeta)$. Instead sample $c_{\mathsf{inv}} \xleftarrow{\$} \mathsf{DJ.Enc}_N(0)$. This is indistinguishable assuming KDM security of Damgård-Jurik.

$\mathcal{H}_5$. Replace the encryptions $x_i^{(1)} \xleftarrow{\$} \mathsf{DJ.Enc}_N(x)$ with encryptions of zero $x_i^{(1)} \xleftarrow{\$} \mathsf{DJ.Enc}_N(0)$. This is indistinguishable by CPA security of Damgård-Jurik (which is implied by KDM security). The adversary's view now only depends on the semi-private inputs y_i, so this hybrid is a valid simulator for the HSS security game.

A.2 From DCR

In this section, we prove theorem 22. That is, we show that fig. 9 is a Semi-Private HSS scheme (definition 16) with linear offline/online sharing of semi-private inputs. We prove each property of the Offline-Online HSS separately.

A.2.1 Correctness. Let $\lambda \in \mathbb{N}$ be a security parameter and $B \leq 2^{(\zeta - 1 - \varepsilon) \cdot \lambda_{\mathsf{DCR}} / 2}$ a bound, let C be an $n_{\mathsf{s\text{-}priv}}$-input circuit over $\mathbb{Z}$ of pebbling depth D, let C_{rm} be a restricted-multiplication n_{priv}-input circuit over $\mathbb{Z}$, let $(y_{\mathsf{s\text{-}priv},i})_{i \in [n_{\mathsf{s\text{-}priv}}]}$ and $(x_{\mathsf{priv},i})_{i \in [n_{\mathsf{priv}}]}$ be B-admissible inputs for C and C_{rm} respectively.

Let us prove that correctness (definition 16) holds with all but negligible probability. The proof boils down to showing that all of the comments in the pseudocode of fig. 9 (*e.g.* "$\mathsf{sh}^{\mathsf{auth}}_{y_{\mathsf{s\text{-}priv}},0}$ and $\mathsf{sh}^{\mathsf{auth}}_{y_{\mathsf{s\text{-}priv}},1}$ are substractive shares of $k_0 \cdot y_{\mathsf{s\text{-}priv}}$ over $\mathbb{Z}/N^\zeta\mathbb{Z}$") are invariants holding with all but negligible probability.

Idealised PRF. For simplicity, we first consider an idealised version of the HSS scheme of fig. 9 where PRF is replaced with a truly random function (*i.e.* the evaluation keys contain the same polynomial-size random string—because PRF has a polynomial-sized domain—instead of the short PRF key). We will later remove this restriction by a reduction to the security of PRF.

Consider the following random variables: (where $\mathsf{sh}^{\mathsf{auth}}_{|C|+|C_{\mathsf{rm}}|,0}$ and $\mathsf{sh}^{\mathsf{auth}}_{|C|+|C_{\mathsf{rm}}|,1}$ are random variables implicitly defined by Eval)

1. $(\mathsf{sk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$;
2. $(x^{(0)}_{\mathsf{priv},i}, x^{(1)}_{\mathsf{priv},i}) \xleftarrow{\$} \mathsf{Share}_{\mathsf{priv}}(1^\lambda, \mathsf{sk}, x_{\mathsf{priv},i})$;
3. $x^{(0)}_{\mathsf{s\text{-}priv},i} \xleftarrow{\$} \mathsf{Share}_{0,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk})$;
4. $x^{(1)}_{\mathsf{s\text{-}priv},i} \xleftarrow{\$} \mathsf{Share}_{1,\mathsf{s\text{-}priv}}(1^\lambda, \mathsf{sk}, y_{\mathsf{s\text{-}priv},i}, \mathsf{st})$;
5. $y_0 \xleftarrow{\$} \mathsf{Eval}(0, \mathsf{ek}_0, (x^{(0)}_{\mathsf{priv},i})^{n_{\mathsf{priv}}}_{i=1}, (y^{(0)}_{\mathsf{s\text{-}priv},i})^{n_{\mathsf{s\text{-}priv}}}_{i=1}, C)$;
6. $y_1 \xleftarrow{\$} \mathsf{Eval}(1, \mathsf{ek}_1, (x^{(1)}_{\mathsf{priv},i})^{n_{\mathsf{priv}}}_{i=1}, (y^{(1)}_{\mathsf{s\text{-}priv},i})^{n_{\mathsf{s\text{-}priv}}}_{i=1}, (C, C_{\mathsf{rm}}))$;
7. $y \xleftarrow{\$} \mathsf{Rec}(y_0, y_1)$

Observe that if we show that $\Pr[y = C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot C_{\mathsf{rm}}(\vec{x}_{\mathsf{priv}})] \geq 1 - \mathsf{negl}(\lambda)$ then we will have proven correctness.

Further consider all the random variables introduced by Eval (we identify the algorithmic variables and the corresponding random variables), and consider the following events:

$$E_{\mathsf{cst}}: \qquad \text{``}\mathsf{sh}^{\mathsf{auth}}_{1,1} - \mathsf{sh}^{\mathsf{auth}}_{1,0} = k_0 \cdot 1\text{''}$$

$$E^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i}: \text{``}\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,1} - \mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,0} = k_0 \cdot y_{\mathsf{s\text{-}priv},i}\text{''} \qquad \text{(for } i \in [n_{\mathsf{s\text{-}priv}}])$$

$$E^{\mathsf{auth}}_i: \qquad \text{``}\mathsf{sh}^{\mathsf{auth}}_{i,1} - \mathsf{sh}^{\mathsf{auth}}_{i,0} = k_{d_i} \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}})\text{''} \qquad \text{(for } i \in [|C|])$$

$$E'_i: \qquad \text{``}\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} = C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_i(\vec{x}_{\mathsf{priv}})\text{''} \qquad \text{(for } i \in [|C_{\mathsf{rm}}|])$$

$$E'_{i,\mathsf{auth}}: \qquad \text{``}\mathsf{sh}^{\mathsf{auth}}_{|C|+i,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+i,0} = k_{d'_i} \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot Q_i(\vec{x}_{\mathsf{priv}})\text{''} \text{ (for } i \in [|C_{\mathsf{rm}}|])$$

where for $i \in [n_{\mathsf{s\text{-}priv}}]$, P_i is the polynomial computed by the ith gate of C (and whose pebbling depth we denote d_i), while for $i \in [n_{\mathsf{priv}}]$, Q_i is the polynomial computed by the ith gate of C_{rm} (and whose pebbling depth we denote d'_i).

Let $C_+ \subseteq [|C|]$ denote the set of indices of the addition gates of C (*i.e.* $i \in C_+$ iff g_i is an addition). For all $i \in C_+$ we consider the events $E^{\mathsf{KS}}_{i,L}$ and $E^{\mathsf{KS}}_{i,R}$

$$E^{\mathsf{KS}}_{i,L}: \text{``}\mathsf{sh}^{\mathsf{auth},d_i}_{i,L,1} - \mathsf{sh}^{\mathsf{auth},d_i}_{i,L,0} = k_{d_i} \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})\text{''}$$

$$E^{\mathsf{KS}}_{i,R}: \text{``}\mathsf{sh}^{\mathsf{auth},d_i}_{i,R,1} - \mathsf{sh}^{\mathsf{auth},d_i}_{i,R,0} = k_{d_i} \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})\text{''}$$

where i_L and i_R are the indices of the parents of gate g_i, and P_i, P_{i_L}, P_{i_R} are the polynomials

computed by gates g_i, g_{i_L}, g_{i_R} (recall that d_i denotes the pebbling depth of gate g_i).

Let $C_\times \subseteq [|C|]$ denote the set of indices of the multiplication gates of C (i.e. $i \in C_\times$ iff g_i is a multiplication). For all $i \in C_\times$ we consider the events $E_{i,L}^{\mathsf{KS}}$ and $E_{i,R}^{\mathsf{KS}}$

$$E_{i,L}^{\mathsf{KS}}: \text{``}\mathsf{sh}_{i,L,1}^{\mathsf{auth},d_i-1} - \mathsf{sh}_{i,L,0}^{\mathsf{auth},d_i-1} = k_{d_i-1} \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})\text{''}$$

$$E_{i,R}^{\mathsf{KS}}: \text{``}\mathsf{sh}_{i,R,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i,R,0}^{\mathsf{auth},d_i} \quad = k_{d_i} \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})\text{''}$$

where i_L and i_R are the indices of the parents of gate g_i, and P_i, P_{i_L}, P_{i_R} are the polynomials computed by gates g_i, g_{i_L}, g_{i_R} (recall that d_i denotes the pebbling depth of gate g_i).

Lemma 50. $\Pr[E_{\mathsf{cst}}] = 1$.

Lemma 36 follows immediately from the definition of $\mathsf{sh}_{0,1}^{\mathsf{auth}}$ and $\mathsf{sh}_{0,1}^{\mathsf{auth}}$.

Lemma 51. $\forall i \in [n_{\mathsf{s\text{-}priv}}], \Pr[E_{\mathsf{s\text{-}priv},i}^{\mathsf{auth}}] \geq 1 - N^{-\varepsilon}$.

Proof of Lemma 51. By definition of the $\mathsf{sh}_{\mathsf{s\text{-}priv},i,1}^{\mathsf{auth}}$ ($i \in [n_{\mathsf{s\text{-}priv}}]$) the following relation holds with probability 1:

$$\forall i \in [n_{\mathsf{s\text{-}priv}}], \mathsf{sh}_{\mathsf{s\text{-}priv},i,1}^{\mathsf{auth}} - \mathsf{sh}_{\mathsf{s\text{-}priv},i,0}^{\mathsf{auth}} \equiv k_0 \cdot y_{\mathsf{s\text{-}priv},i} \bmod N^\zeta$$

Because the offline inputs are all B-admissible, and because $k_0 \cdot B \cdot B \leq N \cdot B = N^{\zeta-\varepsilon} < < N^\zeta$, we get the desired results by invoking Lemma 9 (because the $\mathsf{sh}_{\mathsf{s\text{-}priv},i,0}^{\mathsf{auth}}$, $\mathsf{sh}_{\mathsf{priv},j,0}$, and $\mathsf{sh}_{\mathsf{priv},j,0}^{\mathsf{auth}}$ are sampled uniformly at random from $\mathbb{Z}/N^\zeta\mathbb{Z}$). $\square$

Lemma 52. *Let $i \in [|C|]$. If g_i (the ith gate of C) is the constant gate then $\Pr[E_i^{\mathsf{auth}}] = 1$.*

Proof of Lemma 52. If g_i is the constant gate 1, then by definition $E_i^{\mathsf{auth}} \equiv E_{\mathsf{cst}}$ so $\Pr[E_i^{\mathsf{auth}}] = 1$ by Lemma 50. $\square$

Lemma 53. *Let $i \in [|C|]$. If g_i (the ith gate of C) is the jth input then $\Pr[E_i^{\mathsf{auth}}|E_{\mathsf{s\text{-}priv},j}^{\mathsf{auth}}] = 1$.*

Proof of Lemma 53. If g_i is the jth input gate $y_{\mathsf{s\text{-}priv},j}$, then by definition $E_i^{\mathsf{auth}} \equiv E_{\mathsf{s\text{-}priv},j}^{\mathsf{auth}}$ so $\Pr[E_i^{\mathsf{auth}}|E_{\mathsf{s\text{-}priv},j}^{\mathsf{auth}}] = 1$ because any event is certain when conditioned on itself. $\square$

Lemma 54. *Let $i \in [|C|]$. If g_i (the ith gate of C) is a scalar multiplication with parent g_j, then $\Pr[E_i^{\mathsf{auth}}|E_j^{\mathsf{auth}}] = 1$.*

Proof of Lemma 54. If g_i is the scalar multiplication $\cdot_\alpha$ with predecessor g_j, note that by definition of the pebbling depth g_j and g_i have the same pebbling depth d_i. Therefore, conditioned on E_j:

$$\begin{aligned} \mathsf{sh}_{i,1}^{\mathsf{auth}} - \mathsf{sh}_{i,0}^{\mathsf{auth}} &= \alpha \cdot \mathsf{sh}_{j,1}^{\mathsf{auth}} - \alpha \cdot \mathsf{sh}_{j,0}^{\mathsf{auth}} \\ &= \alpha \cdot (\mathsf{sh}_{j,1}^{\mathsf{auth}} - \mathsf{sh}_{j,0}^{\mathsf{auth}}) \\ &= \alpha \cdot k_{d_i} \cdot P_j(\vec{y}_{\mathsf{s\text{-}priv}}) \\ &= k_{d_i} \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}}) \end{aligned}$$

Therefore $\Pr[E_i^{\mathsf{auth}}|E_j^{\mathsf{auth}}] = 1$. $\square$

Lemma 55. *For all $i \in C_+$, if the parents of (the addition) g_i are g_{i_L} and g_{i_R}, $\Pr[E_{i,L}^{\mathsf{KS}}|E_{i_L}^{\mathsf{auth}}] \geq 1 - (d_i - d_{i_L}) \cdot N^{-\varepsilon}$ and $\Pr[E_{i,R}^{\mathsf{KS}}|E_{i_R}^{\mathsf{auth}}] \geq 1 - (d_i - d_{i_R}) \cdot N^{-\varepsilon}$ (where i_L and i_R are the indices of the parents of multiplication g_i, and d_i, d_{i_L}, d_{i_R} are the pebbling depths of g_i, g_{i_L}, g_{i_R}).*

Proof of Lemma 55. Let $i \in C_+$, and let i_L and i_R be the indices of the parents of addition g_i. Let d_i, d_{i_L}, d_{i_R} be the pebbling depths of g_i, g_{i_L}, g_{i_R}. By definition, $\mathsf{sh}_{i_L,1} - \mathsf{sh}_{i_L,0} = P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})$ and $\mathsf{sh}_{i_R,1} - \mathsf{sh}_{i_R,0} = P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})$, so, by Lemma 21:

$$\Pr\left[\mathsf{sh}_{i_L,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i_L,0}^{\mathsf{auth},d_i} = k_{d_i} \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})|E_{i_L}^{\mathsf{auth}}\right] \geq 1 - (d_i - d_{i_L}) \cdot N^{-\varepsilon}$$

$$\Pr\left[\mathsf{sh}_{i_R,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i_R,0}^{\mathsf{auth},d_i} = k_{d_i} \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})|E_{i_R}^{\mathsf{auth}}\right] \geq 1 - (d_i - d_{i_R}) \cdot N^{-\varepsilon} \ .$$

$\square$

Lemma 56. *For all $i \in C_+$, $\Pr[E_i^{\mathsf{auth}}|E_{i,L}^{\mathsf{KS}}, E_{i,R}^{\mathsf{KS}}] = 1$.*

Proof of Lemma 56. Let $i \in C_+$. If $E_{i,L}^{\mathsf{KS}}$ and $E_{i,R}^{\mathsf{KS}}$ then

$$\begin{aligned}
\mathsf{sh}_{i,1}^{\mathsf{auth}} - \mathsf{sh}_{i,0}^{\mathsf{auth}} &= (\mathsf{sh}_{i_R,1}^{\mathsf{auth},d_i} + \mathsf{sh}_{i_L,1}^{\mathsf{auth},d_i}) - (\mathsf{sh}_{i_R,0}^{\mathsf{auth},d_i} + \mathsf{sh}_{i_L,0}^{\mathsf{auth},d_i}) \\
&= (\mathsf{sh}_{i_L,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i_L,0}^{\mathsf{auth},d_i}) + (\mathsf{sh}_{i_R,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i_R,0}^{\mathsf{auth},d_i}) \\
&= k_{d_i} \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}}) + k_{d_i} \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}}) \\
&= k_{d_i} \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}})
\end{aligned}$$

$\square$

Lemma 57. *For all $i \in C_\times$, $\Pr[E_{i,L}^{\mathsf{KS}}|E_{i_L}^{\mathsf{auth}}] \geq 1 - (d_i - d_{i_L} - 1) \cdot N^{-\varepsilon}$ and $\Pr[E_{i,R}^{\mathsf{KS}}|E_{i_R}^{\mathsf{auth}}] \geq 1 - (d_i - d_{i_R}) \cdot N^{-\varepsilon}$ (where i_L and i_R are the indices of the parents of multiplication g_i, and d_i, d_{i_L}, d_{i_R} are the pebbling depths of g_i, g_{i_L}, g_{i_R}).*

Proof of Lemma 57. By definition, $\mathsf{sh}_{i_L,1} - \mathsf{sh}_{i_L,0} = P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})$ and $\mathsf{sh}_{i_R,1} - \mathsf{sh}_{i_R,0} = P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})$, so, by Lemma 21:

$$\Pr\left[\mathsf{sh}_{i_L,1}^{\mathsf{auth},d_i-1} - \mathsf{sh}_{i_L,0}^{\mathsf{auth},d_i-1} = k_{d_i-1} \cdot P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}})|E_{i_L}^{\mathsf{auth}}\right] \geq 1 - (d_i - 1 - d_{i_L}) \cdot N^{-\varepsilon}$$

$$\Pr\left[\mathsf{sh}_{i_R,1}^{\mathsf{auth},d_i} - \mathsf{sh}_{i_R,0}^{\mathsf{auth},d_i} = k_{d_i} \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})|E_{i_R}^{\mathsf{auth}}\right] \geq 1 - (d_i - d_{i_R}) \cdot N^{-\varepsilon} \ .$$

$\square$

Lemma 58. *For all $i \in C_\times$, $\Pr[E_i|E_{i,L}^{\mathsf{KS}}, E_{i,R}^{\mathsf{KS}}, E_{\mathsf{corr}}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 58. Let $i \in [|C|]$ such that g_i is a multiplication gate whose two operands are g_{i_L} and g_{i_R} (with $i_L, i_R < i$). Denote P_{i_L} and P_{i_R} the polynomials computed by g_{i_L} and g_{i_R} respectively. By core lemma 1, conditioned on $E_{i_L}^{\mathsf{KS}}$, $E_{i_R}^{\mathsf{KS}}$, and E_{corr}:

$$\begin{aligned}
\Bigg[- \mathsf{DDLog}\bigg((c_{d_i-1})^{P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}})} &+ (h_i)^{P_{i_L}(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot \mathsf{sh}_{i,R,1}^{\mathsf{auth},d_i}} \\
+ (h_{i+1})^{P_{i_R}(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot \mathsf{sh}_{i,L,1}^{\mathsf{auth},d_i-1}} &+ g^{-\mathsf{sh}_{i,L,1}^{\mathsf{auth},d_i-1} \cdot \mathsf{sh}_{i,R,1}^{\mathsf{auth},d_i}} \bigg) \Bigg] \\
- \left[-\mathsf{DDLog}\left(g^{-\mathsf{sh}_{i,L,0}^{\mathsf{auth},d_i-1} \cdot \mathsf{sh}_{i,R,0}^{\mathsf{auth},d_i}} \right) \right] &\equiv k_{d_i} \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}}) \bmod N^\zeta
\end{aligned}$$

We conclude, by Lemma 9 (because $k_{d_i} \cdot P_i(\vec{y}_{\mathsf{s\text{-}priv}}) \leq N \cdot B \leq N^{\zeta-\varepsilon}$), that $\Pr\left[E_i|E_{i_L}^{\mathsf{KS}}, E_{i_R}^{\mathsf{KS}}, E_{\mathsf{corr}}\right] \geq 1 - N^{-\varepsilon}$. $\square$

Lemma 59. *Let $i \in [|C_{\mathsf{rm}}|]$. If g_i' (the ith gate of C_{rm}) is the constant gate 1, then $\Pr[E_i'] = 1$ and $\Pr[E_{i,\mathsf{auth}}'|E_{|C|}] = 1$.*

Lemma 59 follows immediately from the definitions of $\mathsf{sh}_{|C|+i,1}$, $\mathsf{sh}_{|C|+i,0}$, $\mathsf{sh}^{\mathsf{auth}}_{|C|+i,1}$, and $\mathsf{sh}^{\mathsf{auth}}_{|C|+i,0}$.

Lemma 60. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is the addition of gates g'_{i_L} and g'_{i_R}, then $\Pr[E'_i | E'_{i_L}, E'_{i_R}] = 1$ and $\Pr[E'_{i,\mathsf{auth}} | E'_{i_L,\mathsf{auth}}, E'_{i_R,\mathsf{auth}}] = 1$.*

Proof of Lemma 60. Conditioned on E'_{i_L} and E'_{i_R},

$$
\begin{aligned}
\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} &= (\mathsf{sh}_{|C|+i_L,1} + \mathsf{sh}_{|C|+i_R,1}) - (\mathsf{sh}_{|C|+i_L,0} + \mathsf{sh}_{|C|+i_R,0}) \\
&= (\mathsf{sh}_{|C|+i_L,1} - \mathsf{sh}_{|C|+i_L,0}) + (\mathsf{sh}_{|C|+i_R,1} - \mathsf{sh}_{|C|+i_R,0}) \\
&= C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_L}(\vec{x}_{\mathsf{priv}}) + C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}}) \\
&= C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot (P_{i_L} + P_{i_R})(\vec{x}_{\mathsf{priv}}) \\
&= C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_i(\vec{x}_{\mathsf{priv}})
\end{aligned}
$$

and therefore $\Pr[E'_i | E'_{i_L}, E'_{i_R}] = 1$.
Similarly, conditioned on $E'_{i_L,\mathsf{auth}}$ and $E'_{i_R,\mathsf{auth}}$,

$$
\begin{aligned}
\mathsf{sh}^{\mathsf{auth}}_{|C|+i,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+i,0} &= (\mathsf{sh}^{\mathsf{auth}}_{|C|+i_L,1} + \mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,1}) - (\mathsf{sh}^{\mathsf{auth}}_{|C|+i_L,0} + \mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,0}) \\
&= (\mathsf{sh}^{\mathsf{auth}}_{|C|+i_L,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+i_L,0}) + (\mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+i_R,0}) \\
&= k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_L}(\vec{x}_{\mathsf{priv}}) + k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}}) \\
&= k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot (P_{i_L} + P_{i_R})(\vec{x}_{\mathsf{priv}}) \\
&= k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_i(\vec{x}_{\mathsf{priv}})
\end{aligned}
$$

and therefore $\Pr[E'_{i,\mathsf{auth}} | E'_{i_L,\mathsf{auth}}, E'_{i_R,\mathsf{auth}}] = 1$. □

Lemma 61. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is a scalar multiplication gate with parent g'_j, then $\Pr[E'_i | E'_j] = 1$ and $\Pr[E'_{i,\mathsf{auth}} | E'_j] = 1$.*

Proof of Lemma 61. Conditioned on E'_j,

$$
\begin{aligned}
\mathsf{sh}_{|C|+i,1} - \mathsf{sh}_{|C|+i,0} &= (\alpha \cdot \mathsf{sh}_{|C|+j,1}) - (\alpha \cdot \mathsf{sh}_{|C|+j,0}) \\
&= \alpha \cdot (\mathsf{sh}_{|C|+j,1} - \mathsf{sh}_{|C|+j,0}) \\
&= \alpha \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_j(\vec{x}_{\mathsf{priv}}) \\
&= C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot (\alpha \cdot P_j)(\vec{x}_{\mathsf{priv}}) \\
&= C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_i(\vec{x}_{\mathsf{priv}})
\end{aligned}
$$

and therefore $\Pr[E'_i | E'_{i_L}, E'_{i_R}] = 1$.
Similarly, conditioned on $E'_{j,\mathsf{auth}}$,

$$
\begin{aligned}
\mathsf{sh}^{\mathsf{auth}}_{|C|+i,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+i,0} &= (\alpha \cdot \mathsf{sh}^{\mathsf{auth}}_{|C|+j,1}) - (\alpha \cdot \mathsf{sh}^{\mathsf{auth}}_{|C|+j,0}) \\
&= \alpha \cdot (\mathsf{sh}^{\mathsf{auth}}_{|C|+j,1} - \mathsf{sh}^{\mathsf{auth}}_{|C|+j,0}) \\
&= \alpha \cdot k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_j(\vec{x}_{\mathsf{priv}}) \\
&= k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot (\alpha \cdot P_j)(\vec{x}_{\mathsf{priv}}) \\
&= k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_i(\vec{x}_{\mathsf{priv}})
\end{aligned}
$$

and therefore $\Pr[E'_{i,\mathsf{auth}} | E'_{j,\mathsf{auth}}] = 1$. □

Lemma 62. *Let $i \in [|C_{\mathsf{rm}}|]$. If g'_i (the ith gate of C_{rm}) is the jth input gate, then $\Pr[E'_i | E_{|C|}] \geq 1 - N^{-\varepsilon}$ and $\Pr[E'_{i,\mathsf{auth}} | E_{|C|}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 62. By Lemma 8, conditioned on $E_{|C|}$,

$$\mathsf{DDLog}((c_{j,0}^{\mathsf{sh}_{|C|,1}^{\mathsf{auth}}}) \cdot (c_{j,1}^{\mathsf{sh}_{|C|,1}^{\mathsf{auth}}})) - \mathsf{DDLog}((c_{j,0}^{\mathsf{sh}_{|C|,0}^{\mathsf{auth}}}) \cdot (c_{j,1}^{\mathsf{sh}_{|C|,0}^{\mathsf{auth}}})) \equiv C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j} \bmod [N^\zeta]$$

and

$$\mathsf{DDLog}(((c_{j,0}^{\mathsf{auth}})^{\mathsf{sh}_{|C|,1}^{\mathsf{auth}}}) \cdot ((c_{j,1}^{\mathsf{auth}})^{\mathsf{sh}_{|C|,1}^{\mathsf{auth}}}))$$
$$- \mathsf{DDLog}(((c_{j,0}^{\mathsf{auth}})^{\mathsf{sh}_{|C|,0}^{\mathsf{auth}}}) \cdot ((c_{j,1}^{\mathsf{auth}})^{\mathsf{sh}_{|C|,0}^{\mathsf{auth}}})) \equiv k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j} \bmod [N^\zeta]$$

Because $C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j}$, $k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot x_{\mathsf{priv},j} \leq N \cdot B^2 \leq N^{\zeta-\varepsilon}$, the desired results therefore follow from Lemma 9. $\qquad\square$

Lemma 63. *Let $i \in [|C_{\mathsf{rm}}|]$. If g_i' (the ith gate of C_{rm}) is a multiplication of the jth input gate and gate g_{i_R}', then $\Pr[E_i'|E_j', E_{|C|+i_R}] \geq 1 - N^{-\varepsilon}$ and $\Pr[E_{i,\mathsf{auth}}'|E_j', E_{|C|+i_R}] \geq 1 - N^{-\varepsilon}$.*

Proof of Lemma 63. By Lemma 8, conditioned on $E_{|C|+i_R}$,

$$\mathsf{DDLog}((c_{j,0}^{\mathsf{sh}_{|C|+i_R,1}^{\mathsf{auth}}}) \cdot (c_{j,1}^{\mathsf{sh}_{|C|+i_R,1}^{\mathsf{auth}}})) - \mathsf{DDLog}((c_{j,0}^{\mathsf{sh}_{|C|+i_R,0}^{\mathsf{auth}}}) \cdot (c_{j,1}^{\mathsf{sh}_{|C|+i_R,0}^{\mathsf{auth}}})) \equiv C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}}) \bmod [N^\zeta]$$

and

$$\mathsf{DDLog}(((c_{j,0}^{\mathsf{auth}})^{\mathsf{sh}_{|C|+i_R,1}^{\mathsf{auth}}}) \cdot ((c_{j,1}^{\mathsf{auth}})^{\mathsf{sh}_{|C|+i_R,1}^{\mathsf{auth}}}))$$
$$- \mathsf{DDLog}(((c_{j,0}^{\mathsf{auth}})^{\mathsf{sh}_{|C|+i_R,0}^{\mathsf{auth}}}) \cdot ((c_{j,1}^{\mathsf{auth}})^{\mathsf{sh}_{|C|+i_R,0}^{\mathsf{auth}}})) \equiv k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}}) \bmod [N^\zeta]$$

Because $C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}})$, $k_D \cdot C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot P_{i_R}(\vec{x}_{\mathsf{priv}}) \leq N \cdot B^2 \leq N^{\zeta-\varepsilon}$, the desired results therefore follow from Lemma 9. $\qquad\square$

Lemma 64. *Correctness with an idealised PRF (i.e. the event "$y = C(\vec{y}_{\mathsf{s\text{-}priv}}) \cdot C_{\mathsf{rm}}(\vec{x}_{\mathsf{priv}})$") holds with all but negligible probability.*

Proof of Lemma 64. The proof is completely analogous to that of Lemma 48. We first rename/order the events $E_{\mathsf{cst}}, E_{\mathsf{s\text{-}priv},1}^{\mathsf{auth}}, \ldots, E_{\mathsf{s\text{-}priv},n_{\mathsf{s\text{-}priv}}}^{\mathsf{auth}}, E_1^{\mathsf{auth}}, \ldots, E_{|C|}^{\mathsf{auth}}, E_1', E_{1,\mathsf{auth}}', \ldots, E_{|C_{\mathsf{rm}}|}', E_{|C_{\mathsf{rm}}|,\mathsf{auth}}'$ as $A_1, \ldots, A_{n_{\mathsf{s\text{-}priv}}+|C|+2|C_{\mathsf{rm}}|+1}$.

$$\Pr[E_{|C_{\mathsf{rm}}|}'] \geq 1 - \sum_{i=1}^{n_{\mathsf{s\text{-}priv}}+|C|+2|C_{\mathsf{rm}}|+1} \Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j]$$

Note that:

- If $i = 1$, $\Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j]$ is negligible by Lemma 50.
- If $2 \leq i \leq n_{\mathsf{s\text{-}priv}} + 1$, $\Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j]$ is negligible by Lemma 51.
- If $2 + n_{\mathsf{s\text{-}priv}} + 1 \leq n \leq 2 + n_{\mathsf{s\text{-}priv}} + |C|$, $\Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j]$ is negligible by Lemmata 52 to 58 (the precise lemmata to be invoked depend on the nature of the ith gate of C).
- If $2 + n_{\mathsf{s\text{-}priv}} + |C| + 1 \leq n \leq 2 + n_{\mathsf{s\text{-}priv}} + |C| + 2|C_{\mathsf{rm}}|$, $\Pr[\overline{A_i} \cap \bigcap_{1 \leq j < i} A_j]$ is negligible by Lemmata 59 to 63 (the precise lemmata to be invoked depend on the nature of the ith gate of C_{rm}).

Therefore there is a negligible function negl such that $\Pr[E_{|C_{\mathsf{rm}}|,\mathsf{auth}}'] \geq 1 - \mathsf{negl}$. $\qquad\square$

Real PRF. We now no longer assume that PRF is instantiated as a truly random function. Consider the adversary which, tasked with distinguishing outputs of PRF and a random oracle, runs the "HSS correctness experiment" and outputs 1 if and only if correctness holds. By security of PRF, the advantage of this adversary can be at most $\varepsilon_{\mathsf{PRF}}$, and since we have already established that correctness (respectively authenticated correctness) in the ideal world holds with probability at least $p \cdot (1 - N^{-\varepsilon})$ (respectively p), where $p = 1 - \mathsf{negl}$, it follows that correctness (respectively authenticated correctness) in the real world holds with all but negligible probability.

A.2.2 Key-switching security.

Lemma 20. *Let* $N \xleftarrow{\$} \mathsf{RSA.Gen}(1^\lambda)$, $g \xleftarrow{\$} (\mathbb{Z}/N^{\zeta+1}\mathbb{Z})^\times$, $x, y \xleftarrow{\$} [0, N)$ *and* $r \xleftarrow{\$} [0, N^\zeta)$. *Then, the distributions*

$$(N, g, g^x, g^y, g^{xy} \cdot \exp(y)) \qquad and \qquad (N, g, g^x, g^y, g^{xy} \cdot \exp(r))$$

are computationally indistinguishable under the DCR assumption.

Proof. We give a hybrid proof, starting from the real distribution $(N, g, g^x, g^y, g^{xy} \cdot \exp(y))$.

$\mathcal{H}_1$. Let $g_N = g^{N^\zeta}$. Use DCR to replace g with g_N in the distribution. [5]

$\mathcal{H}_2$. Sample $x \xleftarrow{\$} [0, 2^\lambda N^{\zeta+1})$ instead of $x \xleftarrow{\$} [0, N)$. This is indistinguishable because x is only used in the exponent of g_N, which has order dividing φ, and both distributions have $x \bmod \varphi$ statistically indistinguishable from uniform. $x \xleftarrow{\$} [0, N)$ can be divided into $x \in [0, \varphi)$, where $x \bmod \varphi$ is uniform, and $x \in [\varphi, N)$, which has negligible probability $\frac{N-\varphi}{N} < \frac{p+q}{N} = O(N^{-\frac{1}{2}})$. Here, p and q are the prime factors of N, which are roughly of order $N^{\frac{1}{2}}$. Similarly, $x \xleftarrow{\$} [0, 2^\lambda N^{\zeta+1})$ can be divided into $x \in [0, 2^\lambda N^{\zeta+1} - (2^\lambda N^{\zeta+1} \bmod \varphi))$, where $x \bmod \varphi$ is uniform, and $x \in [2^\lambda N^{\zeta+1} - (2^\lambda N^{\zeta+1} \bmod \varphi), 2^\lambda N^{\zeta+1})$, which has negligible probability less than $\frac{\varphi}{2^\lambda N^{\zeta+1}} < 2^{-\lambda}$.

$\mathcal{H}_3$. Set $g_N = g^{N^\zeta} \exp(1)$. By DCR, both distributions for g_N are indistinguishable from uniform.

$\mathcal{H}_4$. Let $d = \varphi(\varphi^{-1} \bmod N^\zeta)$, and replace x with $x - d$. This has statistical distance $\frac{d}{2^\lambda N^{\zeta+1}} < 2^{-secpar}$. We can then simplify using

$$g_N^{x-d} = g_N^x g_N^{-d} = g_N^x (g^{N^\zeta} \exp(1))^{-d} = g_N^x \exp(-d) = g_N^x \exp(-1),$$

since g^{N^ζ} has order dividing $\varphi \mid d$, and $d \equiv 1 \bmod N^\zeta$. Therefore, $g_N^{(x-d)y} \cdot \exp(y) = g_N^{xy} \cdot \exp(-y) \cdot \exp(y) = g_N^{xy}$. The distribution is now $(N, g_N, g_N^x \exp(-1), g_N^y, g_N^{xy})$.

$\mathcal{H}_5$. Use DCR to set g_N back to g^{N^ζ}.

$\mathcal{H}_6$. Sample $y \xleftarrow{\$} [0, 2^\lambda N^{\zeta+1})$ instead of $y \xleftarrow{\$} [0, N)$. This is indistinguishable by the same argument as $\mathcal{H}_2$.

$\mathcal{H}_7$. Use DCR to again set g_N to $g^{N^\zeta} \exp(1)$.

$\mathcal{H}_8$. Replace x by $x + d$, and simplify using $g_N^{x+d} = g_N^x \exp(1)$. This is statistically indistinguishable, same as $\mathcal{H}_4$. The distribution is now $(N, g_N, g_N^x, g_N^y, g_N^{xy} \exp(y))$.

$\mathcal{H}_9$. DCR to again set g_N back to g^{N^ζ}. We're essentially back at $\mathcal{H}_1$, except that x and y are now sampled uniformly from $[0, 2^\lambda N^{\zeta+1})$ instead of $[0, N)$.

---

[5] While this is only directly DCR when $\zeta = 1$, the assumption is equivalent to a version where ζ is any $\mathsf{poly}(\lambda)$. [DJ01]

$\mathcal{H}_{10}$. y is used in two places: in the exponent of g_N^y, and in $\exp(y)$. The former only depends on $y \bmod \varphi$, while the latter only depends on $y \bmod N^\zeta$. Since $y \bmod \varphi N^\zeta$ has statistical distance at most $\frac{\varphi N^\zeta}{2^\lambda N^{\zeta+1}} < 2^{-\lambda}$ from uniform, we can use CRT to replace y with two variables: $y \overset{\$}{\leftarrow} [0, \varphi)$ used for g_N^y, and $r \overset{\$}{\leftarrow} N^\zeta$ used for $\exp(r)$. The distribution is now $(N, g_N, g_N^x, g_N^y, g_N^{xy} \exp(r))$.

$\mathcal{H}_{11}$. Replace $x \overset{\$}{\leftarrow} [0, 2^\lambda N^{\zeta+1})$ and $y \overset{\$}{\leftarrow} [0, \varphi)$ by $x, y \overset{\$}{\leftarrow} [0, N)$. This is indistinguishable by a similar argument to $\mathcal{H}_1$.

$\mathcal{H}_{12}$. Use DCR to change replace g_N with g. We are now at the random distribution. $\qquad \square$

A.2.3   Security against party $\sigma = 0$. We need to simulate the view of party $\sigma = 0$ using just the numbers $n_{\text{priv}}, n_{\text{s-priv}}$. We present a sequnece of hybrids, starting from the real world, and ending in world where the view of party 0 will depend only on these values. This last world becomes the simulation.

$\mathcal{H}_1$. This is the real world. The adversary sees $\mathsf{ek}_0 \leftarrow (N, g, (h_i)_{i=1}^D, (c_i)_{i=0}^{D-1}, \mathsf{sh}_{1,0}^{\mathsf{auth}}, k_{\mathsf{PRF}}), x_i^{(0)}$ (output by $\mathsf{Share}_{\text{priv}}$) for $i \in [n_{\text{priv}}]$, and $y_i^{(0)}$ (output by $\mathsf{Share}_{1,\text{s-priv}}$) for $i \in n_{\text{s-priv}}$. Note that $\mathsf{HSS.Share}_{0,\text{s-priv}}$ is not provided y_i as an input, so the only thing that depends on private information is the ciphertexts $x_i^{(0)} = (c_{x_i}, c_{x_i}^{\mathsf{auth}})$, so this is what we must show can be simulated.

$\mathcal{H}_2$. Perform a sequence of hybrids $\mathcal{H}_2^0 \ldots, \mathcal{H}_2^D$. Let $\mathcal{H}_2^0 = \mathcal{H}_1$ be the real world and let $\mathcal{H}_2 = \mathcal{H}_2^D$. In the change from $\mathcal{H}_2^i$ to $\mathcal{H}_2^{i+1}$, we replace $c_i \leftarrow g^{k_i k_{i+1}} \cdot \exp(k_{i+1})$ with $c_i \leftarrow g^{k_i k_{i+1}} \cdot \exp(r_{i+1})$, where $r_{i+1} \overset{\$}{\leftarrow} [0, N^\zeta)$. Finally, in $\mathcal{H}_2^D = \mathcal{H}_2$ we have set every $c_i = g^{k_i k_{i+1}} \cdot \exp(r_{i+1})$.

We show for all i that the change from $\mathcal{H}_2^i$ to $\mathcal{H}_2^{i+1}$ is indistinguishable, by reduction to Lemma 20. Let $A = g^x, B = g^y, C = g^{xy} \exp(y)$ in the challenge from Lemma 20. Set $c_i \leftarrow C$, $h_i = A$, and $h_{i+1} = B$. Sample all k_j for $j \notin \{i, i+1\}$ as normal, and compute h_j as normal. Also compute c_j for $j \notin [i-1, i+1]$ as normal, using r_{j+1} when $j < i$. Finally, we have to compute c_{i-1} and c_{i+1}, which depend on the challenge keys $x = k_i$ and $y = k_{i+1}$. We can compute these just given A and B: $c_{i-1} = A^{k_{i-1}} \exp(r_i)$ and $c_{i+1} = B^{k_{i+2}} \exp(k_{i+2})$. These are the only places where k_i and k_{i+1} are used in the adversary's view. Therefore, a distinguisher for $\mathcal{H}_2^i$ and $\mathcal{H}_2^{i+1}$ implies a distinguisher for the real and random distributions in Lemma 20.

$\mathcal{H}_3$. Now, the only use of k_D in the adversry view is to compute the public keys $h_D = g^{k_D}$ and $c_{D-1} = g^{k_{D-1} k_D} \exp(r_D)$, and to compute the private share ciphertexts $c_{x_i}^{\mathsf{auth}} \overset{\$}{\leftarrow} \mathsf{DJE.Enc}_{\mathsf{pk}_D}(k_D \cdot x_i)$. Note that h_D is a standard Damgård-Jurik-ElGamal public key, while c_{D-1} can be computed from h_D, k_{D-1}, and $\exp(r_D)$. Therefore, we can use KDM security (theorem 6) to replace $c_{x_i}^{\mathsf{auth}} \overset{\$}{\leftarrow} \mathsf{DJE.Enc}_{\mathsf{pk}_D}(k_D \cdot x)$ with encryptions of uniformly random plaintexts. Simultainously, we replace $c_{x_i} \overset{\$}{\leftarrow} \mathsf{DJE.Enc}_{\mathsf{pk}_D}(x_i)$ with more encryptions of a random plaintexts.

The distribution is now independent of all private inputs x_i, and indeed of everything the HSS security simulator is not given. Therefore, we can view the adversary's view in this hybrid as the simulator.

A.2.4   Semi-private security against party $\sigma = 1$. We need to simulate the view of party $\sigma = 1$ using just the numbers $n_{\text{priv}}, n_{\text{s-priv}}$, and the values of the semi-private shares y_i. We present a sequnece of hybrids, starting from the real world, and ending in world where the view of party 1 will depend only on these values. This last world becomes the simulation.

$\mathcal{H}_1$. This is the real world. The adversary sees $\mathsf{ek}_1 \leftarrow (N, g, (h_i)_{i=1}^D, (c_i)_{i=0}^{D-1}, \mathsf{sh}_{1,1}^{\mathsf{auth}}, k_{\mathsf{PRF}}), x_i^{(1)}$

(output by $\mathsf{Share}_{\mathsf{priv}}$) for $i \in [n_{\mathsf{priv}}]$, and $y_i^{(1)} = (y_i, \mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,1})$ (output by $\mathsf{Share}_{1,\mathsf{s\text{-}priv}}$) for $i \in n_{\mathsf{s\text{-}priv}}$.

$\mathcal{H}_2$. Instead of sampling $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,0} \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}$ in $\mathsf{HSS.Share}_{0,\mathsf{s\text{-}priv}}$ and setting $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,1}) \leftarrow (k_0 \cdot y_i + \mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,0}) \bmod N^\zeta$ in $\mathsf{HSS.Share}_{1,\mathsf{s\text{-}priv}}$, sample $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,1} \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}$ and set $\mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,0}) \leftarrow (-k_0 \cdot y_i + \mathsf{sh}^{\mathsf{auth}}_{\mathsf{s\text{-}priv},i,0}) \bmod N^\zeta$. This is an identical distribution.

$\mathcal{H}_3$. Make a similar change to $\mathsf{sh}^{\mathsf{auth}}_{1,0}$ and $\mathsf{sh}^{\mathsf{auth}}_{1,1}$. Instead of sampling $\mathsf{sh}^{\mathsf{auth}}_{1,0} \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}$ and setting $\mathsf{sh}^{\mathsf{auth}}_{1,1} \leftarrow (k_0 + \mathsf{sh}^{\mathsf{auth}}_{1,0}) \bmod N^\zeta$, sample $\mathsf{sh}^{\mathsf{auth}}_{1,1} \xleftarrow{\$} \mathbb{Z}/N^\zeta \mathbb{Z}$ and set $\mathsf{sh}^{\mathsf{auth}}_{1,0} \leftarrow (-k_0 + \mathsf{sh}^{\mathsf{auth}}_{1,0}) \bmod N^\zeta$. Again, this is an identical distribution.

$\mathcal{H}_4$. Notice that the adversary's view (other than y_j) is essentially the same as it would if party $\sigma = 0$ were corrupted instead. That is, the real distribution for ek_0 is the same as the previous hybrid's distribution for ek_1, the private input ciphertexts are equal always $(x_i^{(0)} = x_i^{(1)})$, and the real distribution for $\mathsf{sh}_{\mathsf{s\text{-}priv},i,0}$ is the same as the previous hybrid's distribution for $\mathsf{sh}_{\mathsf{s\text{-}priv},i,1}$. Therefore, we can replace all of these by a simulation that depends on only n_{priv} and $n_{\mathsf{s\text{-}priv}}$, by security against $\sigma = 0$ proven in the previous section. Putting these together with the semi-private inputs y_i revealed to the simulator, this gives a simulation of the $\sigma = 1$ adversary's view.

B Deferred Material on Masking

We now present two instantiations of admissible masking schemes for our garbling scheme.

B.0.1 Boolean Truth Tables First, we consider arbitrary boolean gates with $\ell = O(\log(s))$ inputs, i.e., truth tables of size $\mathsf{poly}(s)$. We now show that a simple XOR masking scheme is admissible with these wide-gates.

$$\mathcal{K} = \mathcal{X} = \mathcal{Y} = \{0,1\} \qquad \mathsf{Mask}_k(x) = k \oplus x \qquad \mathsf{UnMask}_k(y) = k \oplus y \qquad (4)$$

Lemma 65. *The XOR masking scheme (eq. (4)) and size $\mathsf{poly}(s)$ boolean truth tables are admissible for $\mathsf{RMS} \otimes \mathsf{VP}$.*

Proof. Need to be able to evaluate a masked version of any arbitrary truth table $T[x_1, \ldots, x_\ell]$ on $\ell = O(\log(s))$ bits. That is, we need to write

$$(k_1, \ldots, k_\ell, k', y_1, \ldots, y_\ell) \mapsto T[k_1 \oplus y_1, \ldots, k_\ell \oplus y_\ell] \oplus k'$$

as an $\mathsf{RMS} \otimes \mathsf{VP}$ circuit over $\mathbb{Z}$:

$$\sum_{i=0}^{2^\ell - 1} F_i(k_1, \ldots, k_\ell, k') G_i(y_1, \ldots, y_\ell).$$

We do this by setting

$$F_i = T[k_1 \oplus i_1, \ldots, k_\ell \oplus i_\ell] \oplus k' \qquad G_i = (1 \oplus i_1 \oplus y_1) \cdots (1 \oplus i_\ell \oplus y_\ell),$$

where $(i_1, \ldots, i_\ell)$ are the bits of the ℓ-bit integer i. That is, F_i computes the masked truth table on input $y = i$, while G_i computes 1 if $x = i$ and 0 otherwise. Here, we have used that any truth table on logarithmically many inputs can be evaluated by a polynomial-sized RMS program. $\qquad \square$

B.0.2 Multivariate Polynomials Next, we consider rate-1 garbling of arithmetic circuits with plaintext values in some bounded subset $[-B, B]$ of $\mathbb{Z}$, à la [MORS24]. Since both the HSS and arithmetic circuit are defined over $\mathbb{Z}$, we use additive masking over $\mathbb{Z}$.

$$
\begin{aligned}
\mathcal{K} &= [B, B + 2B(2^\lambda - 1)] \\
\mathcal{X} &= [-B, B] & \mathsf{Mask}_k(x) &= x + k \\
\mathcal{Y} &= [0, 2B\, 2^\lambda] & \mathsf{UnMask}_k(y) &= y - k
\end{aligned}
\tag{5}
$$

Because $\mathbb{Z}$ is infinite, we have to make the set of masks $\mathcal{K}$ much wider than the set of plaintext values $\mathcal{X}$, and even then we only get statistical privacy. Note that while the masked values are bigger than the unmasked ones, the rate $\frac{\log_2(B)}{\log_2(B)+\lambda}$ approaches 1 if $\log_2 = \omega(\lambda)$.

Lemma 66. *Additive masking (eq. (5)) is a one-time private masking scheme (definition 24).*

Proof. Let the simulator sample $y \xleftarrow{\$} \mathcal{Y}$. For all $x \in [-B, B]$, the only time when y is more likely to be sampled as $y \xleftarrow{\$} \mathcal{Y}$ than as $y \leftarrow \mathsf{Mask}_k(x)$ is when $y \notin \mathcal{K} + x$. Therefore, the statistical distance from the simulation is

$$
\Pr_{y \xleftarrow{\$} \mathcal{Y}} [y \notin \mathcal{K} + x] = \frac{2B2^\lambda + 1 - 2B(2^\lambda - 1) - 1}{2B2^\lambda + 1} = \frac{2B}{2B2^\lambda + 1} < 2^{-\lambda}.
$$

$\square$

Next, we must show admissibility together with a class of wide-gates. Following [Cou19, BCM23], we choose our wide-gate class to be $\log(s)$-degree multivariate polynomials that can be written explicitly with $\mathsf{poly}(s)$ monomials.

Lemma 67. *Additive masking, together with the class of gates that can evaluate any polynomial p with degree $\log(s)$ and $\mathsf{poly}(s)$ monomials, is admissible for $\mathsf{RMS} \otimes \mathsf{VP}$. That is,*

$$
(k_1, \dots, k_\ell, k', y_1, \dots, y_\ell) \mapsto T[y_1 - k_1, \dots, y_\ell - k_\ell] + k'
$$

can be written as an $\mathsf{RMS} \otimes \mathsf{VP}$ circuit

$$
\sum_{i=1}^{r} F_i(k_1, \dots, k_\ell, k') G_i(y_1, \dots, y_\ell)
$$

for some circuits F_i, G_i, and $r = \mathsf{poly}(s)$.

Proof. Because in the end we get to sum over polynomially many terms, we only need to prove the case where p is a single monomial $x_{i_1} x_{i_2} \cdots x_{i_{\log(s)}}$.[6] Setting $x_{i_j} = \mathsf{UnMask}_{k_{i_j}}(y_{i_j})$ and expanding, we get

$$
x_{i_1} x_{i_2} \cdots x_{i_{\log(s)}} = \sum_{t_1, \dots, t_{\log(s)} \in \{0,1\}} (-k_{i_1})^{1 - t_{i_1}} \cdots (-k_{i_{\log(s)}})^{1 - t_{\log(s)}} \cdot y_{i_1}^{t_{i_1}} \cdots y_{i_{\log(s)}}^{t_{\log(s)}}.
$$

This is a sum of $2^{\log(s)} = \mathsf{poly}(s)$ terms, and each is a product of a monomial in k and a monomial in y. Therefore, p can be written as an $\mathsf{RMS} \otimes \mathsf{VP}$ circuit. $\square$

---

[6] With the indices allowed to repeat.