Hybrid Zero-Knowledge from Garbled Circuits*[⋆]*

Masayuki Abe¹, Miguel Ambrona², and Miyako Ohkubo³

¹NTT Laboratories, abe.masayuki@iecl.ntt.co.jp, ² Midnight, miguel.ambrona@iohk.io ³ NICT, m.ohkubo@nict.go.jp

We present techniques for constructing zero-knowledge argument systems from garbled circuits, extending the GC-to-ZK compiler by Jawurek, Kerschbaum, and Orlandi (ACM CCS 2023) and the GC-to-*Σ* compiler by Hazay and Venkitasubramaniam (J. Crypto, 2020) to the following directions:

- Our schemes are *hybrid, commit-and-prove* zero-knowledge argument systems that establish a connection between secrets embedded in *algebraic* commitments and a relation represented by a *Boolean* circuit.
- Our schemes incorporate diverse *cross-domain* secrets embedded within distinct algebraic commitments, simultaneously supporting Pedersen-like commitments and lattice-based commitments.

As an application, we develop circuit-represented compositions of *Σ*-protocols that support attractive access structures, such as weighted thresholds, that can be easily represented by a small circuit. For predicates P_1, \ldots, P_n individually associated with a *Σ*-protocol, and a predicate *C* represented by a Boolean circuit, we construct a *Σ*-protocol for proving $C(P_1, \ldots, P_n) = 1$. This result answers positively an open question posed by Abe, et. al., at TCC 2021.

Keywords: hybrid zero-knowledge, garbled circuit, *Σ*-protocol, composition

[⋆] Preliminary version of this paper was presented at SCN2024. This is the full version.

Table of Contents

1 Introduction

Garbled circuits (GC) [\[49,](#page-25-0) [50\]](#page-25-1) are a fundamental building block for constructing various cryptographic protocols. Zero-knowledge proof systems are no exception, as they can be seen as a particular case of two-party computation where only one party (the prover) has an input (the witness), and the other party (the verifier) obtains the result of evaluating the concerned relation on the input. Several works, e.g.,[\[40,](#page-24-0) [31,](#page-24-1) [37,](#page-24-2) [19,](#page-23-0) [33\]](#page-24-3), have been devoted to constructing ZK using GC as a black box. Due to the significant communication overhead per gate in garbling, ZK from GC best suits proving relations that can be expressed by small circuits. Nevertheless, their modular structure enables attractive yet often overlooked properties such as online-offline delayed inputs, low-latency pipelining, and small on-memory processing. It also facilitates the understanding of algorithms, lowering the barrier for development and maintenance[\[27,](#page-24-4) [16\]](#page-23-1). These properties are orthogonal to those of other design paradigms, such as succinct non-interactive arguments of knowledge (SNARKs) [\[35\]](#page-24-5), which have seen significant advances in recent years due to demand from blockchain applications.

There are two approaches of ZK from GC in the literature. One is represented by [\[40\]](#page-24-0) where the verifier garbles the circuit and the prover evaluates it with the witness. The garbling is privacy-free and is done only once. Despite its interactive nature for obliviously transmitting encoding keys, its value is demonstrated in practical scenarios, e.g., [\[46\]](#page-25-2). The other approach introduced in [\[37,](#page-24-2) [27\]](#page-24-4) is the opposite; the prover garbles the circuit and the verifier evaluates it with a given encoded witness. As their construction follows the cut-and-choose strategy to ensure the correctness of garbling, several garbling and evaluations must be done. Instead, it results in standard *Σ*-protocols [\[23\]](#page-23-2), which is a significant advantage in generic protocol designs. In particular, it can be non-interactive through the Fiat-Shamir transformation. Nevertheless, its practical potential remains unexplored.

Commit-and-prove ZK is a practical extension of zero-knowledge systems for bridging several proofs on common inputs by reusing the same commitment of the witness. Hybrid ZK is a particular class of commit-and-prove ZK whose commitments are algebraic, but a Boolean circuit represents the relation to prove. Consider two commitments $com(s)$ and $com(t)$ for secrets *s* and *t* for which the prover argues that both the *i*-th bit of *s* and the *j*-th bit of *t* are 1. A tiny circuit with just one AND gate can represent such a relation. However, enforcing consistency between the input bits to the circuit and the committed secrets could become a cumbersome task. This is particularly challenging with *cross-domain secrets*, i.e., where the commitments are based on different domains and mechanisms, like Pedersen commitments over elliptic-curve groups and lattice-based commitments. (See Section [1.2](#page-6-0) for more discussion.)

1.1 Our Contribution

The goal of this work is to extend the ZK-from-GC paradigms of $[40]$ and $[37]$ ^{[1](#page-3-1)} to construct hybrid, commit-and-prove zero-knowledge argument systems that establish a connection between a relation represented by a *Boolean* circuit and secrets embedded in *algebraic* commitments. They incorporate diverse *crossdomain* secrets embedded within a broad range of algebraic commitments.

Our technical results are organized as follows:

- We first introduce a new garbling scheme supporting Boolean and affine operations over rings altogether, generalizing the garbling technique of [\[34\]](#page-24-6) (Section [4\)](#page-13-0). It is used to solve the input consistency issue for connecting a Boolean circuit and algebraic commitments. This garbling is used commonly for all constructions in this paper.
- The garbling scheme is then used to extend the construction of universally composable zero-knowledge from garbled circuits in [\[40\]](#page-24-0) (ZKGC hereafter) to their commit-and-prove variant (CP-ZKGC) (Section [5.2\)](#page-15-2). We show that CP-ZKGC allows monotone compositions for proving partial knowledge (Section [5.3\)](#page-16-0).
- We next build a *cut-and-choose garbled-circuit* (CCZK) compiler. It converts a privacy-preserving garbling scheme into a Sigma protocol following the MPC-in-the-Head with pre-processing paradigm [\[41\]](#page-24-7) (Section [6.1\)](#page-17-1). It is then combined with our garbling scheme from Section [4](#page-13-0) to get its commit-and-prove variant (CP-CCZK). The resulting CP-CCZK supports monotone composition, and provides non-interactive hybrid ZK from GC via the Fiat-Shamir transformation. For the CCZK compiler to work efficiently, we introduce a new building block, a shuffled label commitment scheme (Appendix [B\)](#page-29-0), that improves the communication complexity induced by the cut-and-choose structure of CCZK. Overall efficiency improvement from a naive construction is the 90% saving in the proof size.
- As an application of CP-CCZK, we present a general circuit-represented composition of *Σ*-protocols (Section [7\)](#page-22-0). Given *Σ*-protocols for proving arbitrary predicates P_1, \ldots, P_n and a garbling scheme for Boolean circuit C , we construct a *Σ*-protocol for predicate $C(P_1, \ldots, P_n)$. Such a composition brings a solution for attractive relations such as *weighted threshold proofs* where committed secrets are assigned some weights, and circuit *C* verifies whether the proven knowledge's total weight exceeds a threshold. Note that the circuit can use non-monotone gates, but the function that it represents must be monotone.^{[2](#page-3-2)} This is a qualitative improvement over previous work [\[25\]](#page-24-8), given

¹ We refer to the public-coin honest verifier ZK in Section 6 of [\[37\]](#page-24-2). They also present interactive ZK with commit-and-prove property based on the 2PC-in-the-Head paradigm.

 $\frac{2}{3}$. This is inherent. Note that a modular composition of statements under non-monotone functions is not a well-defined notion, as the prover could always pretend not to know one of the witnesses.

that monotone span programs (or even monotone circuits) are not universal for efficient computation of monotone functions, see, e.g., $[47]$, whereas non-monotone circuits are.

We review our technical contributions slightly in more details below.

Definitional Work. In Section [3,](#page-10-1) we introduce refined security notions for garbling schemes focused on authenticity and verifiability, aiding in identifying necessary and sufficient properties for our constructions.

The standard authenticity is a computational problem that it is hard for the adversary to compute a correct output label. For the purpose of composition where the output from a circuit is used in further computation, we propose a decisional variant of authenticity, which we call *output indistinguishably*. This property enables replacing unrevealed output parts with random values independent of the garbled circuit, useful for reductions involving other connected components. It is achieved by applying a suitable key derivation function to the output labels.

Correctness typically permits negligible error probability, but in cut-andchoose protocols, this may allow adversaries to choose coins so that evaluation fails. We thus need it to work perfectly *once verified*. We refine verifiability in [\[40,](#page-24-0) [31\]](#page-24-1) to closely align with correctness, and introduce a variation *verifiable correctness* as required in our cut-and-choose construction, CCZK. While standard constructions readily achieve these notions, articulating them helps avoid pitfalls when relaxed versions are desired.

Algebraic Commitments and Affine Garbling. We consider a class of commitment schemes whose commitment function com is a *generalized affine function*, which consists of an affine transformation over a ring or a transformation from a ring to its module. To compute such a function in a garbled manner, we introduce an *affine garbling scheme*, which is a garbling scheme for generalized affine functions, and prove several properties as defined as above.

Wire Merging. Different circuit types use specific wire labeling strategies for efficiency. A challenge arises when different types of circuits share the inputs. To address this, we devised a garbling scheme to merge wire labels seamlessly, ensuring consistency of committed values and inputs throughout computation. We extend the wire merging technique of $[34]$ developed for privacy-free garbling by leveraging the color-bit technique [\[14\]](#page-23-3) to add privacy as needed in CCZK.

Cut-and-Choose with Preprocessing. Our CCZK compiler follows the *MPC-inthe-Head with pre-processing* paradigm, also known as KKW [\[41\]](#page-24-7) as outlined below.

(Pre-processing phase.)

1. The prover plays as a trusted party in each session by garbling the circuit, and committing to the input wire label pairs generated, shuffling each pair (through the *shuffled label commitment scheme* as explained later). The prover executes *M* sessions and sends the verifier all the garbled circuits and commitments.

- 2. The verifier selects τ sessions randomly as a challenge.
- 3. For each non-selected session, the prover publishes the garbled circuit and the whole input wire labels in the correct order. This can be done efficiently by publishing the random coins used for garbling.
- 4. The verifier accepts if, for all but τ sessions, the garbled circuits and the encoding keys are correctly formed.

(On-line phase.)

- 5. For every selected session, the prover publishes the garbled circuit and the encoded witness \hat{x} represented by half of the input wire labels and the selective opening information *π*.
- 6. The verifier accepts if the commitment is recovered from \hat{x} and π , and \hat{x} evaluates the garbled circuit to 1 for all the selected τ sessions.

The protocol can be made three rounds by merging steps 3 and 4 with and 5 and 6, respectively.

Unlike KKW, cheating during the on-line phase is rendered impossible if the circuits are honestly garbled. The soundness error is upper-bounded essentially by the probability of guessing τ out of *M* sessions. With $M = 512$ and $\tau =$ 22 for 128-bit security, this optimization saves the number of evaluated GCs down to $22/64 = 34\%$ compared to a naive construction with $M = 128$ and $\tau = 64$. Including the savings by our shuffled label commitment, the dominant communication of sending encoded inputs is reduced to 20%.

Shuffled Label Commitments. We develop an efficient shuffled label commitment scheme (Appendix [B.4\)](#page-32-0) that functions like a vector commitment with either all or a random half of the labels authentically opened. We present an efficient construction that combines a Merkle tree [\[44\]](#page-25-4) and an NNL length-doubling pseudo-random generator [\[45\]](#page-25-5). Our analysis shows that it reduces the average opening size to 61%, which is a dominant factor in the proof size.

While the concept is simple, the construction requires careful techniques. The challenge is that labels in algebraic circuits can be very short, and revealing a hash value of these short labels as a sibling can compromise the security of the undisclosed labels. To address this, we devised steps to introduce pseudo-entropy without significantly increasing the size of the disclosed information, making the formal security proof complex. We prove security based on the pseudo-randomness of the NNL generator and the *MT correlation robustness* of the Merkle hash (Definition [15\)](#page-27-1).

Circuit-Represented Composition. Our idea is to incorporate a flag *sⁱ* directly indicating the prover's possession of the witness for atomic predicate P_i . For proving a composed statement, $C(P_1, \ldots, P_n)$, the prover does as follows:

- 1. Commit to (s_1, \ldots, s_n) .
- 2. Prove $(P_i = 1 \land s_i = 1) \lor (s_i = 0)$ for every $i \in [n]$.
- 3. Prove $C(s_1, \ldots, s_n) = 1$.

We use hybrid zero-knowledge for proving $C(s_1, \ldots, s_n) = 1$ on committed secrets (s_1, \ldots, s_n) . This way, composition *C* can be represented by a Boolean circuit. The proof in the second step associates flag s_i with predicate P_i . Provided a *Σ*-protocol for proving $s_i \in \{0,1\}$, it can be done through a standard composition method [\[26\]](#page-24-9).

1.2 Related Works

Comparison to hybrid ZKGC in [\[19\]](#page-23-0). Our advantage of CP-ZKGC over [\[19\]](#page-23-0) is primarily qualitative; the generality of the supported commitments and the extended functionality allowing monotone composition. Regarding the performance, we compare the communication and computation for the example of proving statement $\{(m, r) | \text{com}(m; r) = c \land C(m)\}\$ for a Boolean predicate *C*, with labels of size λ_1 and com being Pedersen commitment with group size λ_2 . Let $COT(a, b)$ denote the bandwidth for committed OT with *a* choice bits and *b* total input message size. Let \hat{C} denote the garbled C . The dominant term in the $\text{bandwidth of } [19] \text{ is } |m|\lambda_2 + \text{COT}(|m|, 2\lambda_1|m|) + |m|\lambda_2 + |\hat{C}| + H_1(\lambda_1) + H_2(|m| +$ $\text{bandwidth of } [19] \text{ is } |m|\lambda_2 + \text{COT}(|m|, 2\lambda_1|m|) + |m|\lambda_2 + |\hat{C}| + H_1(\lambda_1) + H_2(|m| +$ $\text{bandwidth of } [19] \text{ is } |m|\lambda_2 + \text{COT}(|m|, 2\lambda_1|m|) + |m|\lambda_2 + |\hat{C}| + H_1(\lambda_1) + H_2(|m| +$ $|m|\lambda_2 + |r| + \lambda_2^2$ + $\Pi_3(|m| + 2\lambda_2 + \lambda_2^2)$ where $\Pi_i(a)$ denotes the bandwidth for the *i*-th zero-knowledge proof whose witness size is *a*. For CP-ZKGC, it is $\text{COT}(|m|+|r|, 2\lambda_1(|m|+|r|))+|\tilde{C}|+\lambda_2|m|$. With a simple oblivious transfer from [\[20\]](#page-23-4) over 256-bit elliptic-curve groups, we estimate that [\[19\]](#page-23-0) requires 140 kBytes of communication additionally to $|\hat{C}|$, whereas it is 66 kBytes with CP-ZKGC. The prover computes 3335 exponentiations in [\[19\]](#page-23-0) that is reduced to 513 with CP-ZKGC.

Relation to arithmetic garbling in [\[34\]](#page-24-6), [\[12\]](#page-23-5), and more. Our affine garbling is a mixture of techniques from [\[12\]](#page-23-5) and [\[34\]](#page-24-6). We follow [\[34\]](#page-24-6) in our affine garbling with three substantial differences. The first is generality. We extend their approach to which we call generalized affine operations over arbitrary rings, supporting lattice-based cryptography. The second is that our garbling protects privacy, which is essential for our construction of non-interactive zero-knowledge. The third is the presence of a formal composition theorem with a Boolean garbling, which is missing in their work. In $[12]$, a binary decomposed value over a ring is encoded in the same manner as ours. They support high-degree arithmetic operations with inefficient decoding, which is not required in our case for garbling affine predicates.

The encoding method used in SPDZ- 2^k [\[24,](#page-24-10) [13\]](#page-23-6) has a similar flavor. They extend the domain to 2^{s+k} with sufficiently large *s* to achieve demanded soundness error 2^{-s} . We instead choose to repeat the garbing for the target domain η times so that $2^{-\eta k}$ suffices for soundness. It simplifies the algorithm, particularly when the arithmetic operations are extended to groups, in exchange for losing fine-tuning in setting the target error level.

On Garbled Circuit to Σ-protocols. Other than the simple cut-and-choose construction mentioned above, [\[37\]](#page-24-2) introduces a paradigm called 2PC-in-the-Head

that builds a *Σ*-protocol from garbled circuits. It is a variation of MPC-in-the-Head [\[39\]](#page-24-11) where the witness is secret-shared among the virtual players who cooperate to evaluate the target function. It inherently doubles the input length of the evaluating circuit, impacting the efficiency, and is used as a theoretical tool in subsequent works, e.g., [\[42\]](#page-25-6).

Hybrid zero-knowledge from other than garbling schemes. There are efficient constructions of interactive ZK, e.g., [\[13,](#page-23-6) [48\]](#page-25-7), for hybrid statements that combine efficient proof systems within Boolean or arithmetic relations using edaBits [\[29\]](#page-24-12) optimized for the zero-knowledge scenario. Those schemes base on subfield vector oblivious linear evaluation (sVOLE, [\[15\]](#page-23-7)), and are particularly well-suited for scenarios that allow an interactive preprocessing phase between the prover and the verifier.

Regarding zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs), there are several works, e.g., $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ $[3, 18, 32, 17, 4]$ that address commitand-prove structure as well as compositions. They are well suited for Pedersen-like commitments in the classical domain.

The MPC-in-the-Head [\[39\]](#page-24-11) (IKOS) paradigm provides alternative ways to construct *Σ*-protocols for proving Boolean or arithmetic relations. Hybrid zeroknowledge from IKOS/KKW paradigms is a potential approach for some specific hybrids. In [\[38\]](#page-24-14), a direct extension of KKW that supports transitions from Boolean and arithmetic representations via edaBits [\[29\]](#page-24-12) is addressed. Since [\[38\]](#page-24-14) encodes binary values without overhead, while CCZK encodes a bit with a *λ*-bit label for security parameter λ , their proof size is λ times smaller. The resulting Σ -protocol inherits a specific soundness property and does not compose straightforwardly.

A series of works, including [\[7,](#page-23-10) [6,](#page-22-3) [9,](#page-23-11) [5\]](#page-22-4), have made significant progress in the compressed *Σ*-protocols, considering various assumptions. These protocols can be categorized as hybrid zero-knowledge since they utilize algebraic commitments and provide proofs of relations represented by Boolean circuits. The underlying computation principle of these protocols relies on the homomorphic property of the commitments and how witnesses are encoded within them.

On general compositions of Σ-protocols. Early works on composition methods [\[26,](#page-24-9) [25\]](#page-24-8) support a wide range of access structures, including monotone formulas and monotone span programs. Subsequent works, such as [\[21,](#page-23-12) [22,](#page-23-13) [1,](#page-22-5) [8,](#page-23-14) [11,](#page-23-15) [30,](#page-24-15) [36\]](#page-24-16), enhance efficiency for specific access structures like disjunctions and thresholds. A recent work [\[2\]](#page-22-6), considers composition represented by acyclicity programming, a new model of computation that is incompatible with monotone span programming. Their method inherently turns *Σ*-protocols into non-interactive arguments in the random oracle model. A composition preserving the structure of sigma protocols with a model of computation that subsumes monotone span programs was an open problem. Our construction indeed produces *Σ*-protocols for desired compound statements.

2 Preliminaries

2.1 Notations

By $a \leftarrow b$, we denote that structured object *b* is parsed into *a*. In case of parse error, the relevant algorithm halts outputting a special symbol ⊥. For positive integers $a, b \in \mathbb{N}$, we denote by [*a*] and [*a*, *b*] the range $\{1, \ldots, a\}$ and ${a, a+1, \ldots, b}$, respectively. For a finite set, *S*, $a \leftarrow S$ denotes uniform sampling from *S*, and |*S*| denotes carnality of *S*. By $(x_i)_{i \in I}$, we denote $(x_{i_1}, \ldots, x_{i_{|I|}})$ for $i_j \in I$. For a ring element *x*, $\langle x \rangle$ denotes binary representation of *x*, and $|x|$ is the number of bits in $\langle x \rangle$. For a binary value $\langle x \rangle := (x_1, \ldots, x_\ell) \in \{0, 1\}^\ell$ and 2 ℓ elements indexed as $(k_i^0, k_i^1)_{i \in [\ell]}$, we denote by k^x an array of ℓ elements $k^x := (k_i^{x_i})_{i \in [\ell]}$. By $\{w : R(w)\}$ we denote a statement that witness *w* satisfies relation *R*. We denote by $\text{PoK}\{w : R(w)\}\$ a proof of knowledge system about witness *w* satisfying relation *R*. Variables not included in *w* are understood as an instance.

For finite set *X*, we denote the number of bits needed to represent an arbitrary element of *X* by $\mathsf{rep}(X)$, and the number of bits needed to represent an array of all elements in *X* by len(*X*). Formally, $\mathsf{rep}(X) := \max_{x \in X}(|x|)$, and len(*X*) := $\sum_{x \in X} |x|$.

2.2 Affine Predicate and Affine Commitment

Let $(\mathcal{R}, +, \cdot)$ be a commutative ring and group $(\mathcal{G}, +)$ be right \mathcal{R} -module with scalar multiplication operation \odot : $\mathcal{G} \times \mathcal{R} \rightarrow \mathcal{G}$. In particular, we use the following properties: for all $a, b \in \mathcal{G}$ and $x, y \in \mathcal{R}$, $(a \odot x) \odot y = a \odot (x \cdot y)$ and $(a + b) \odot x = a \odot x + b \odot x$. We also require that, for fixed $a \neq 0$, $a \odot x$ is injective from R to G, and for all distinct a, b in G and non-zero x in R, $a \odot x \neq b \odot x$. Let $g: \mathcal{R}^u \to \mathcal{R}^v$ be an affine function over \mathcal{R} where $g(x) := g_1 \cdot x + g_2$ for $g_1 \in \mathcal{R}^{v \times u}$ and $g_2 \in \mathcal{R}^v$ for some positive integers *u* and *v* with operations \cdot and $+$ extended to matrix multiplication and vector addition. Similarly, we consider generalized affine function $f: \mathcal{R}^u \to \mathcal{G}^v$ where $f(x) = \hat{g}_1 \odot x + \hat{g}_2$ for $\hat{g}_1 \in \mathcal{G}^{v \times u}$ and $\hat{g}_2 \in \mathcal{G}^v$. For $x \in \mathcal{R}^u$ and its binary representation $\langle x \rangle$ translated into $\{0_{\mathcal{R}}, 1_{\mathcal{R}}\}^{\ell}$, we define bit-composition by $G \cdot \langle x \rangle = x$ where $G \in \mathcal{R}^{u \times \ell}$ is a gadget matrix. We define *affine predicate*, *F*, for generalized affine function *f* and constant *y*. It takes binary representation of *x* and outputs 1 if $f(x) = y$, otherwise outputs 0, i.e., $F(\langle x \rangle) := (f(x) \stackrel{?}{=} y).$

Definition 1 (Affine Commitment Scheme). *A commitment scheme is a tuple of polynomial-time algorithms* (gen*,* com*,* open) *that:*

- $\text{gen}(1^{\lambda}) \rightarrow ck$ *is a commitment-key generation algorithm that, on input security parameter λ, outputs a commitment key ck. The commitment key defines the message space and the randomness space.*
- $\text{com}_{ck}(m;r) \rightarrow c$ *is a commitment function, on input ck, message m, and random coins r, outputs a commitment c.*

• $\text{open}_{ck}(c, m, r) \rightarrow 0/1$ *is a predicate that takes commitment c, message m and random coin r and outputs* 1 *or* 0*.*

It is an affine commitment scheme if, additionally,

- $com_{ck}(m;r)$ *is a generalized affine function on m and r over a ring,*
- *there exists a predicate* domver_{ck} $(m;r)$ *that checks if* m *and* r *are in the respective domains defined by ck, and*
- $\mathsf{open}_{ck}(c,m,r)$ $\mathit{outputs}\ 1$ *if and* $\mathit{only}\ if\ (\mathsf{com}_{ck}(m;r) \stackrel{?}{=} c) \land \mathsf{domver}_{ck}(m,r)\ is$ *true.*

It is correct if, for any sufficiently large λ *, ck* \in Gen(1^{λ})*, and any m and r, satisfying* domver_{ck} (m, r) *,* open_{ck} $(\text{com}_{ck}(m; r), m, r)$ *outputs* 1*.*

We follow standard hiding and binding properties [\[28\]](#page-24-17). For simplicity, we often omit gen and *ck* whenever links between com and open are obvious.

Many commitments of practical interest are affine commitments. We further assume that the predicate $domver_{ck}(m, r)$ can be represented by a small boolean circuit taking binary representations of *m* and *r* as input. Given binary representations, checking the input size being smaller than a public constant is free of communication cost. The same is true for checking that the infinity norm is smaller than 2^k for some positive integer k .

2.3 *Σ***-protocol**

Definition 2 (Σ **-protocol).** *A* Σ -protocol for relation $R(y, x) = 1$ is a three*move public-coin protocol between prover* P *having y, x and the verifier* V *having y. It satisfies special soundness and special honest verifier zero-knowledge that:*

- *- Special soundness : There exists a polynomial time algorithm extractor* E *which is given y* and *two* accepting transcripts (a, c, z) and (a, c', z') for $c \neq c'$, *outputs x* which satisfies $R(y, x) = 1$. This property is also referred to as *2-special soundness.*
- *- Special honest verifier zero-knowledge: There exists a polynomial time simulator that outputs a transcript* (*a, c, z*) *on input y and a random c with the same probability distribution as that produced by the honest interaction between* P *and* V *on the common instance y.*

Following [\[10\]](#page-23-16), we handle arguments based on hardness assumptions by extending the statement to $\{x \mid R(y, x) = 1 \vee (x \text{ solves the hard problem})\}.$ Extractor E outputs x that either satisfies $R(y, x) = 1$ or solves the hard problem with probability 1. This way, we achieve knowledge soundness from special soundness. The instance of the hard problem must not be chosen by the prover but provided as a common reference string (CRS) or generated transparently. In our construction, this corresponds to setting up a collision-resistant hash function, usually hardwired in the protocol. Therefore, our construction yields *Σ*-protocols in the CRS or Transparent model. We will handle the setup implicitly in this paper.

3 Garbling Scheme

In this section, we explore several security notions of garbling schemes used in succeeding sections.

Definition 3 (Garbling Scheme [\[14\]](#page-23-3)). *A projective garbling scheme is a triple* $of PPT algorithms \tG = (Gb, Ev, En, De) where:$

- $Gb(1^{\lambda}, C) \rightarrow (\hat{C}, e, d)$: On input security parameter λ and the description of a *function C over domain* $\{0,1\}^{\ell}$ *, outputs a garbled function* \hat{C} *, encoding key e, and a decoding key d.*
- $\mathsf{En}(e, x) \to \hat{x}$: On input encoding key *e* and input value *x*, it outputs an encoded *input value x*ˆ*.*
- $\mathsf{Ev}(\hat{C}, \hat{x}) \to \hat{z}$: On input a garbled \hat{C} and an encoded input \hat{x} , outputs an *encoded output z*ˆ*.*
- De $(\hat{z}, d) \rightarrow z$ *: On input an encoded output* \hat{z} *and a decoding key d, outputs decoded result z.*

It is correct *if for all polynomial-size functions C of domain* $\{0,1\}^{\ell}$ *, and all* $x \in \{0,1\}^{\ell}$, the following probability is negligible in λ :

$$
\Pr\left[\begin{matrix}(\hat{C},e,d)\leftarrow\mathsf{Gb}(1^\lambda,C)\\\hat{x}\leftarrow\mathsf{En}(e,x)\\\hat{z}\leftarrow\mathsf{Ev}(\hat{C},\hat{x})\end{matrix}\right]\colon\mathsf{De}(\hat{z},d)\neq C(x)\right].
$$

We focus on projective garbling schemes where the encoding is applied individually on each input variable. In particular, encoding key *e* defines *input wire label pairs*, $(k_i^0, k_i^1)_{i \in [\ell]}$, and encoding function $\text{En}(e, x)$ outputs $\hat{x} := (k_i^{x_i})_{i \in [\ell]}$. Encoding key *e* is understood as equivalent to the label pairs, but it may be a small random seed for pseudo-random generation of the label pairs. We note that such encoding is invertible in a sense that, given \hat{x} and e , it is easy to obtain x that $\hat{x} = \mathsf{En}(e, x)$ by simple lookup decoding. By En^{-1} , we denote the decoding algorithm. It holds that, for all $e \leftarrow$ Gb₁(1^{\2}, P) and *x* in the domain of P, $En^{-1}(e, En(e, x)) = x$. For any \hat{x} that *x* satisfying $\hat{x} = \text{En}(e, x)$ does not exist, $\text{En}^{-1}(e, \hat{x})$ outputs \perp . We also require a garbling scheme be *separable* in the sense that Gb consists of two algorithms, input label generator Gb_1 and circuit garbler Gb_2 that $e \leftarrow \mathsf{Gb}_1(1^\lambda, C)$ and $(\hat{C}, d) \leftarrow$ Gb₂(*C*, *e*). We further require Gb₁ generate input wire labels in a way that it is perfectly indistinguishable which of each label pair is generated randomly:

Definition 4 (Random Invertible Labels). *Input wire labels are random invertible if there exists an invertible function* D *, and* $k_i^1 = D(k_i^0)$ *holds for each* $pair (k_i^0, k_i^1)$, and either k_i^0 or k_i^1 is chosen (pseudo) randomly.

Labels compatible with Free-Xor, $k_i^1 = k_i^0 \oplus \Delta$, are random invertible. Jumping ahead, labels in our garbling scheme for affine commitments are $k_i^1 = k_i^0 + \delta$ in a ring and are random invertible. The following property holds for separable garbling schemes with random invertible labels.

Lemma 1. *If a separable garbling scheme has random invertible input labels, then, for any circuit C and any input* $x, \hat{x} := \text{En}(\text{Gb}_1(1^{\lambda}, C), x)$ *is indistinguishable from uniform.*

Soundness and verifiability are defined for garbling of predicates. Intuitively, soundness captures the idea that it must be hard to find the encoding of 1, given an encoded input of arbitrary x' such that $f(x') \neq 1$.

Definition 5 (Soundness [\[40\]](#page-24-0)). *A garbling scheme is* sound *if for all polynomialsize predicate* $f: \{0,1\}^{\ell} \to \{0,1\}$, all $x \in \{0,1\}^{\ell}$ that $f(x) = 0$, and all PPT ad*versaries* A, the following probability is negligible in λ :

$$
\Pr\left[(\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^{\lambda}, f) : d \leftarrow \mathcal{A}(\hat{f}, \mathsf{En}(e, x))\right].
$$

We introduce a decisional counterpart of soundness called *output indistinguishability*. For predicates *f*, we only concern output indistinguishability with respect to *x* satisfying $f(x) = 0$.

Definition 6 (Output Indistinguishability). *A garbling scheme is output indistinguishable if there exists a PPT algorithm,* SimD*, that, for all polynomialsize* $f : \{0,1\}^{\ell} \to \{0,1\}^n$, for all PPT adversaries A, and $x \in \{0,1\}^{\ell}$, the *following advantage is negligible in λ:*

$$
\Pr\begin{bmatrix} (\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f) \\ \hat{x} = \mathsf{En}(e, x) \\ b \leftarrow \mathcal{A}(\hat{x}, d) \end{bmatrix} \text{ : } b = 1 \end{bmatrix} - \Pr\begin{bmatrix} (\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^\lambda, f) \\ \hat{x} = \mathsf{En}(e, x) \\ \hat{z} = \mathsf{Ev}(\hat{f}, \hat{x}) \\ d' \leftarrow \mathsf{SimD}(\hat{z}, d) \\ b \leftarrow \mathcal{A}(\hat{x}, d') \end{bmatrix} \text{ : } b = 1 \end{bmatrix}
$$

Privacy and obliviousness capture the idea that encoded input k^x does not give useful information about *x*. Depending on extra information given to the adversary, we use either obliviousness or privacy defined as follows.

Definition 7 (Obliviousness [\[14\]](#page-23-3)). *A garbling scheme is* oblivious *if there exists a polynomial-time simulator* S *that, for all polynomial-size (in* λ) *functions f of domain* $\{0,1\}^{\ell}$ *, and all PPT adversaries* A, the following advantage is upper *bounded by a negligible function in λ:*

$$
\Pr\begin{bmatrix} (x,st) \leftarrow \mathcal{A}(1^{\lambda}) \\ (\hat{f},e,d) \leftarrow \mathsf{Gb}(1^{\lambda},f) \\ \hat{x} := \mathsf{En}(e,x) \\ b \leftarrow \mathcal{A}(st,\hat{f},\hat{x}) \end{bmatrix} \colon b = 1 \right] - \Pr\begin{bmatrix} (x,st) \leftarrow \mathcal{A}(1^{\lambda}) \\ (\hat{f},\hat{x}) \leftarrow \mathcal{S}(1^{\lambda},f) \\ b \leftarrow \mathcal{A}(st,\hat{f},\hat{x}) \end{bmatrix}
$$

Definition 8 (Privacy [\[14\]](#page-23-3)). *A garbling scheme is* private *if there exists a polynomial-time simulator* S *that, for all polynomial-size (in λ) functions f, and all PPT adversaries* A*, the following advantage is upper bounded by a negligible* *function in λ:*

$$
\Pr\begin{bmatrix} (x, st) \leftarrow \mathcal{A}(1^{\lambda}) \\ (\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^{\lambda}, f) \\ \hat{x} \leftarrow \mathsf{En}(e, x) \\ b \leftarrow \mathcal{A}(st, \hat{f}, \hat{x}, d) \end{bmatrix} : b = 1 \right] - \Pr\begin{bmatrix} (x, st) \leftarrow \mathcal{A}(1^{\lambda}) \\ (\hat{f}, \hat{x}, d) \leftarrow \mathcal{S}(1^{\lambda}, f, f(x)) : b = 1 \\ b \leftarrow \mathcal{A}(st, \hat{f}, \hat{x}, d) \end{bmatrix}
$$

We say that privacy simulator S is *separable* if there exists polynomial-time algorithms, S_1 and S_2 , where $S_1(1^\lambda, f, f(x))$ outputs \hat{x} and $S_2(f, f(x), \hat{x})$ outputs \hat{f} and *d*. The notion of privacy may be demanded for inputs belonging to a subset of its domain. For instance, to argue zero-knowledge, we only need to consider privacy over *x* satisfying our target relation (see Section [6.1\)](#page-17-1) and restrict A in the above definition to choose to satisfy *x*.

Often a garbling scheme allows one to extract the decoding key from the whole encoding key and a garbled circuit. As observed in [\[33\]](#page-24-3), this concept is formalized in incomparable flavors as shown in the following definitions. We first present *decode key extractability* where, once the decoding key is extracted, it works perfectly in decoding *any* encoded output.

Definition 9 (Decode Key Extractability). *A garbling scheme is* decode key extractable *if, there exist PPT algorithms,* VeD *and* ExtD*, that, for all polynomial-size (in* λ) functions f of domain $\{0,1\}^{\ell}$, probability

$$
\Pr\left[(\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^{\lambda}, f) \; : \; 1 \neq \mathsf{VeD}(f, \hat{f}, e) \right] \tag{1}
$$

is negligible in λ *, and* $\textsf{ExtD}(\hat{f}, e)$ *outputs d that satisfies* $f(x) = \textsf{De}(\textsf{Ev}(\hat{f}, \textsf{En}(e, x)), d)$ *for all* $x \in \{0,1\}^{\ell}$ *and* (\hat{f}, e, d) *that* $1 = \text{VeD}(f, \hat{f}, e)$ *.*

In [\[40\]](#page-24-0), *verifiability* is defined as slightly limited notion that only concerns output value of 1. This notion suffices for the ZKGC framework in Section [5.](#page-15-0)

Definition 10 (Verifiability). *A garbling scheme is* verifiable *if, there exists PPT algorithms,* VeE *and* ExtE*, that, for all polynomial-size (in λ) functions f of domain* $\{0,1\}^{\ell}$ *, all polynomial-time adversary* A, all $x \in \{0,1\}^{\ell}$ satisfying $f(x) = 1$ *, the following probabilities are negligible in* λ *.*

$$
\Pr\left[(\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^{\lambda}, f) \; : \; 1 \neq \mathsf{VeE}(f, \hat{f}, e) \right] \tag{2}
$$

$$
\Pr\left[\begin{matrix}(\hat{f},e) \leftarrow \mathcal{A}(1^{\lambda},f) \\ 1 \leftarrow \mathsf{VeE}(f,\hat{f},e) \end{matrix}\right] : \mathsf{Ev}(\hat{f},\mathsf{En}(e,x)) \neq \mathsf{ExtE}(\hat{f},e,1)\right]
$$
(3)

Note that it can happen that $\mathsf{Ev}(\hat{f}, k^{\tilde{x}}) = \mathsf{En}(e, 1)$ for \tilde{x} satisfying $f(\tilde{x}) \neq 1$ even if (\hat{f}, e) verifies with VeE. Soundness prevents from finding such \tilde{x} for honestly generated (\hat{f}, e) , but there is no guarantee for maliciously chosen (\hat{f}, e) as in the above definition. Also note that verifiability is implied by decode key extractability, but the reverse is not.

When a garbling scheme is used in the cut-and-choose strategy, it is crucial to verify that the garbling is done honestly. Typically, if a garbling is opened and verified, we expect it to be correct. However, if correctness allows for a small error, the garbling could be manipulated to exploit this. Therefore, once a garbling is opened and verified, it must guarantee perfect correctness. This idea is formalized as *verifiable correctness* as follows and used in our CCZK in Section [6.](#page-17-0)

Definition 11 (Verifiable Correctness). *A garbling scheme is verifiably correct if there exists a polynomial-time algorithm* VeC *that, for all polynomial-size* $function f with domain \{0,1\}^{\ell}, probability$

$$
\Pr\left[(\hat{f}, e, d) \leftarrow \mathsf{Gb}(1^{\lambda}, f) \; : \; 1 \neq \mathsf{VeC}(f, \hat{f}, e, d) \right] \tag{4}
$$

is negligible in λ *, and* $\text{De}(\text{Ev}(\hat{f}, \text{En}(e, x), d) = f(x)$ *holds for all* $x \in \{0, 1\}^{\ell}$ *and* (\hat{f}, e, d) *that* $1 \leftarrow \text{Vec}(f, \hat{f}, e, d)$ *.*

Verifiable correctness is implied by decoding key extractability. But the reverse is not necessarily hold. Note also that soundness and verifiability do not necessarily imply verifiable correctness. They however are achieved in standard garbling schemes.

4 Generalized Affine Garbling with Common Labels

We construct a garbling scheme for affine predicate *F* with a property that some input wires specified by **B-Labels** \subseteq [ℓ] are given labels that can be directly used in garbling other Boolean predicates that share the input wires. The remaining input wires, denoted by A-Labels, are given exclusively to *F*. These attributes are determined arbitrarily and regarded as a part of the specification of *F* in the following description.

In Figure [1,](#page-14-0) we illustrate our garbling scheme whose B-Labels are compatible with Free-XOR garbling [\[43\]](#page-25-8). Note that Ev only computes $f(x) - \hat{g}_2$ and decoding key *d* equals encoding of $y - \hat{g}_2$. It is the decoding function that evaluates the equality. Note also that our scheme does not support high-degree algebraic relations. We prove Theorem [1](#page-13-1) in Appendix [D.1.](#page-36-2)

Theorem [1](#page-14-0). *Algorithms,* (Gb₁, Gb₂, Ev, En, De)*, in Figure 1 constitute a garbling scheme of predicate* $F(x) := (f(x) \stackrel{?}{=} y)$ *for generalized affine function* $f(x) =$ $\hat{g}_1 \odot x + \hat{g}_2$ *and constant y. If* B-Labels = \emptyset *, it is perfectly correct and verifiably correct, verifiable, and it is private with respect to x satisfying* $F(x) = 1$ *. It is output indistinguishable for x satisfying F*(*x*) = 0 *and sound with soundness error* $1/|\mathcal{R}|$ *. If* B-Labels(F) \neq \emptyset *, the same properties hold except for* KDF *be correlation robust for privacy.*

 $\mathsf{Gb}_1(1^\lambda, F)$: Δ ← $\{0, 1\}^{\lambda - 1}$ ||1 $\delta \leftarrow \mathcal{R}^*$ For *i* ∈ B-Labels $k_i^0 \leftarrow \{0,1\}^\lambda, k_i^1 := k_i^0 \oplus \Delta$ For *i* ∈ A-Labels $k_i^0 \leftarrow \mathcal{R}, k_i^1 := k_i^0 + \delta$ Output $e := (k_i^0, k_i^1)_i$. $\mathsf{Gb}_2(F, e)$: $(k_i^0, k_i^1)_i \leftarrow e$ For *i* ∈ B-Labels $\pi_i := \mathsf{LSB}(k^0_i)$ $K_i := \text{KDF}(i, k_i^0), D_i := \text{KDF}(i, k_i^1)$ $t_i := (1 - 2\pi_i)(K_i - D_i + \delta)$ $w_i^0 := (1 - \pi_i) \cdot K_i + \pi_i \cdot (D_i - \delta)$ For *i* ∈ A-Labels $w_i^0 := k_i^0$ *d* := $(\hat{g}_1 \odot (G \cdot w^0)) + ((y - \hat{g}_2) \odot \delta)$ Output $\hat{F} := (t_i)_i$, and *d*. En(*e, x*): $(k_i^0, k_i^1) \leftarrow e$ Output $\hat{x} := k^x$. $\mathsf{Ev}(\hat{F}, \hat{x})$: $\overline{(t_i)_i} \leftarrow \hat{F}, (k_i)_i \leftarrow \hat{x}$ For *i* ∈ B-Labels $\varphi_i := \text{KDF}(i, k_i) + \text{LSB}(k_i) \cdot t_i$ For *i* ∈ A-Labels $\varphi_i := k_i$ $\hat{z} := \hat{g}_1 \odot (\mathsf{G} \cdot \varphi)$ Output *z*ˆ. $\mathsf{De}(\hat{z}, d)$: Output $(\hat{z} \stackrel{?}{=} d)$.

Fig. 1: Garbling affine predicate $F(x) := (\hat{g}_1 \odot x + \hat{g}_2 \stackrel{?}{=} y)$ with partial Free-Xor labels. Index *i* sweeps from 1 to the bit size of *x*. B-Labels and A-Labels determine indices *i* where input wire *i* is with Free-Xor labels or arithmetic ones, respectively. Notation: $w^0 = (w_i^0)_i$, $\varphi = (\varphi_i)_i$. \mathcal{R}^* denotes invertible elements of \mathcal{R} .

Small Domain with Free Range Proof. If the actual inputs to *F* are taken from a smaller domain, say $S_1 \times \cdots \times S_u \subseteq \mathcal{R}^u$ for $\mathsf{rep}(S_i) \leq \mathsf{rep}(\mathcal{R})$, the binary representation of the inputs can be compressed to $n := \sum_{i=1}^{u} \text{rep}(S_i) \le \text{rep}(\mathcal{R}^u)$ by adjusting G so that bit-composition F_G operates over the smaller domain. Thus, the total size of *e* is reduced to $\text{len}(e) = 2\eta \cdot \text{rep}(\mathcal{R}) \cdot \sum_{i=1}^{u} \text{rep}(S_i)$.

This optimization not only saves the space for encoded inputs, but allows range proofs almost for free. Operating on an encoded input that inherently corresponds to one bit, the value is naturally restricted to one or zero. Generally, it extends to range $[0, 2^{\kappa} - 1]$ with κ encoded inputs.

Compact Encoding of Input Bits. When $\mathcal R$ has a subgroup $\mathcal R'$ whose representation is smaller and group operation is the same as that in \mathcal{R} , we can select K_i^b from \mathcal{R}' instead of $\mathcal R$ to gain efficiency. This reduces the total bit size of K by $\mathsf{rep}(\mathcal{R}')/\mathsf{rep}(\mathcal{R}),$ namely, $\mathsf{len}(K) = 2\eta \cdot \mathsf{rep}(\mathcal{R}') \cdot \sum_{i=1}^u \mathsf{rep}(S_i)$ (with the first optimization in mind). This optimization is particularly effective for lattice-based target function *F* where $\mathsf{rep}(\mathcal{R}')/\mathsf{rep}(\mathcal{R}) = |q|/\mathsf{rep}(\mathcal{R})$ is as large as hundreds to a few thousand.

Theorem [1](#page-13-1) holds also for the above optimized construction. We note that the conditions that \mathcal{R}' is a subgroup of $\mathcal R$ and additive operation preserves in \mathcal{R}' and R are sufficient for correctness of GC. Other properties hold almost in the same way with trivial differences on domains.

5 ZKGC: From GC to Interactive UCZK

5.1 Construction

We rely on the following theorem from [\[40\]](#page-24-0) about constructing a zero-knowledge proof system from a projective garbling scheme.

Theorem 2 (Interactive ZK from GC). *[\[40,](#page-24-0) Theorem 2] There exists an efficient compiler that, given a correct, sound, and verifiable projective garbling scheme for a predicate computing a relation, outputs a protocol that securely realizes zero-knowledge proof functionality for the relation in the presence of a malicious prover and a malicious verifier in the* $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{COM}})$ *-hybrid model.*

We refer to [\[40\]](#page-24-0) for concrete constructions substantiating the above theorem. For completeness, we copied it in Appendix [C.](#page-36-0) It is stressed that the ZKGC compiler does not require privacy in the underlying garbling schemes. Privacy-free garbling [\[31,](#page-24-1) [51\]](#page-25-9) produces more compact garbled circuits than privacy-preserving ones. In [\[33\]](#page-24-3), variations of ZKGC that enjoy security against adaptive corruption are presented.

5.2 Extension to Commit-and-Prove: CP-ZKGC

We consider conjunctive composition of Boolean garbling B and arithmetic garblingA for proving relation $P(x) := (C(x) \wedge F(x))$. It is assumed that B and A are *Input Compatible* in the sense that input wire labels and encoding functions are common, i.e., $B.Gb₁ = A.Gb₁$ and $B.En = A.En$. Allowing dedicated inputs to *C* and *F* like $P(x, y, z) := (C(x, y) \wedge F(x, z))$ can be done straightforwardly in the manner by generating input keys separately for *y* and *z*. The resulting U

$U.Gb1(1λ, P)$: $\overline{(C,F)} \leftarrow P$ $e \leftarrow B.Gb_1(1^{\lambda}, C)$	$U.En(e, x)$: $\hat{x} \leftarrow B.En(e, x)$ Output \hat{x} .	$U.De(\hat{z}, d)$: $(\hat{z}_B, \hat{z}_A) \leftarrow \hat{z}$ $(d_{\mathsf{B}}, d_{\mathsf{A}}) \leftarrow d$
Output e .		$b_{\rm B} \leftarrow B \cdot De(\hat{z}_{\rm B} d_{\rm B})$
U.Gb ₂ (P,e): $(C, F) \leftarrow P$ $(\hat{C}, d_{\mathsf{B}}) \leftarrow \mathsf{B.Gb}_2(C, e)$	U.Ev (\hat{P}, \hat{x}) : $(\hat{C}, \hat{F}) \leftarrow \hat{P}$ $\hat{z}_{\text{B}} \leftarrow \text{B.Ev}(\hat{C}, \hat{x})$ $\hat{z}_A \leftarrow A.Ev(\hat{F}, \hat{x})$	$b_{\mathsf{A}} \leftarrow \mathsf{A}.\mathsf{De}(\hat{z}_{\mathsf{A}}d_{\mathsf{A}})$ Output $(b_B \wedge b_A)$?
$(\hat{F}, d_{\mathsf{A}}) \leftarrow \mathsf{A}.\mathsf{Gb}_2(F, e)$ Output $\hat{P} := (\hat{C}, \hat{F})$ and $d := (d_{\mathsf{B}}, d_{\mathsf{A}})$.	Output $\hat{z} := (\hat{z}_{\mathsf{B}}, \hat{z}_{\mathsf{A}}).$	

Fig. 2: Garbling scheme U for conjunctive relation $P(x) := (C(x) \wedge F(x))$.

leaks information about which of B and A fails when $P(x) = 0$. It is, however, irrelevant for our purpose since ZKGC admits privacy-free garbling.

We prove Theorem [3](#page-16-1) in Appendix [D.2.](#page-39-0)

Theorem 3 (Privacy-free Conjunctive Composition). *Garbling scheme* U *in Figure [2](#page-15-3) is correct, sound, and verifiable if both garbling schemes* B *and* A *are correct, sound , and verifiable.*

Applying the ZKGC compiler in Section [5](#page-15-0) to garbling scheme U, we obtain a commit-and-prove zero-knowledge proof protocol for $P(x) = 1$. That constitutes the CP-ZKGC compilation.

5.3 Monotone Compositions of CP-ZKGC

We consider predicates P_1, \ldots, P_n garbled by B or A, and their composition $P = P_0(P_1(x), \ldots, P_n(x))$ where P_0 is a monotone Boolean function representing an access structure. Let SE := (Enc*,* Dec) be a one-time CPA-secure symmetrickey encryption scheme whose key space is $\{0,1\}^{\lambda}$ and message space is $\{0,1\}^*$. Let SS := (Share*,* Rec*,* Ver) be a correct, secure, and verifiable secret sharing scheme. We refer to [A](#page-26-0)ppendix A for these building blocks. For $i \in [n]$, let P_i be a garbling scheme for P_i . All P_i . Gb₁ and P_i . En are common and denoted by Gb¹ and En. Figure [3](#page-16-3) illustrates privacy-free garbling scheme M for predicate *P*. Theorem [4](#page-16-2) is proved in Appendix [D.3.](#page-39-1)

$M.Gb(1^{\lambda}, P)$:	M.Ev (\hat{P}, \hat{x}') :	$M.En(e, x)$:
$(P_0, P_{i \in [n]}) \leftarrow P$	$(\hat{P}_i, c_i)_{i \in [n]} \leftarrow \hat{P}$	$\hat{x} := \mathsf{En}(e, x)$
$e \leftarrow$ Gb ₁ $(1^{\lambda}, P_1)$	$(\hat{x}, x) \leftarrow \hat{x}'$	$\hat{x}' := (\hat{x}, x)$
$\forall i \in [n], (\hat{P}_i, d_i) \leftarrow P_i \cdot \mathsf{Gb}_2(P_i, e)$	$T := \{i \in [n] P_i(x) = 1\}$	Output \hat{x}' .
$d \leftarrow \{0,1\}^{\lambda}$ $t_{i\in[n]} \leftarrow \mathsf{SS}.\mathsf{Share}(P_0, d)$ $\forall i \in [n], c_i \leftarrow \mathsf{SE}.\mathsf{Enc}_{\mathsf{KDF}_i(d_i)}(t_i)$ Output $\hat{P} := (\hat{P}_i, c_i)_{i \in [n]}, e$ and d.	$\forall i \in [T]$ $d_i \leftarrow P_i$. Ev (\hat{P}_i, \hat{x}) $t_i \leftarrow \mathsf{SE}.\mathsf{Dec}_{\mathsf{KDF}_i(d_i)}(c_i)$ $\hat{z} \leftarrow$ SS.Rec $(P_0, t_{i \in [T]})$ Output \hat{z} .	M.De (\hat{z}, d) : Output $\overline{(\hat{z} = d)}$?.

Fig. 3: Privacy-free monotone composition. P_0 : description of monotone access structure over [*n*].

Theorem 4. *Garbling scheme* M *in Figure [3](#page-16-3) is correct, sound, and verifiable* garbling of predicate $P_0(P_1(x), \ldots, P_n(x))$ if every P_i is correct, output indistin*guishable with decoding key domain* D_i *for x satisfying* $P_i(x) = 0$ *, and verifiable,* and SS *is correct, secure, and verifiable,* SE with key space $\{0,1\}^{\lambda}$ *is correct and indistinguishable against chosen plaintext attacks, and every* KDF*ⁱ is indistinguishable over Di.*

Prover(*P*, *x*) Verifier(*P*) $s \leftarrow \{0,1\}^{\lambda}$ $RT := (s, M)$ $(s_i)_{i \in [M]} \leftarrow \mathsf{N}$.Expand (RT) $\forall i \in [M]$ $(e_i, h_i) \leftarrow \mathsf{S}.\mathsf{ComGb}_1(1^\lambda, P; s_i)$ $(\hat{P}_i, d_i) \leftarrow G.Gb_2(P, e_i)$ $H \leftarrow$ Hash $((\hat{P}_i, d_i, h_i)_{i \in [M]})$ \xrightarrow{H} $ST \leftarrow \mathsf{N}.$ SubTrees $(RT, [M] \setminus \mathsf{CH}) \stackrel{\mathsf{CH}}{\longleftarrow} \mathsf{CH} \leftarrow [M]$ ∀*i* ∈ CH $(\tilde{x}_i, \sigma_i) \leftarrow S$ *.ProvEn* (P, x, s_i) $GE := (\hat{P}_i, d_i, \tilde{x}_i, \sigma_i)_{i \in \mathsf{CH}}$ *ST ,GE* $V := [M] \setminus \mathsf{CH}$ $(s_i)_{i \in V} \leftarrow \mathsf{N}$. Expand (ST) ∀*i* ∈ *V* $(e_i, h_i) \leftarrow \mathsf{S}.\mathsf{ComGb}_1(1^\lambda, P; s_i)$ $(\hat{P}_i, d_i) \leftarrow G.Gb_2(P, e_i)$ $1 \stackrel{?}{=}$ G.VeC (P, \hat{P}_i, e_i, d_i) ∀*i* ∈ CH $(\hat{P}_i, d_i, \tilde{x}_i, \sigma_i)_{i \in \text{CH}} \leftarrow GE$ $h_i \leftarrow S.VerCom(\tilde{x}_i, \sigma_i)$ $\hat{x}_i \leftarrow$ S.UnComp (P, \tilde{x}_i) $1 \stackrel{?}{=}$ G.De(G.Ev(\hat{P}_i, \hat{x}_i)*, d_i*) $\mathsf{H} \stackrel{?}{=} \mathsf{Hash}((\hat{P}_i, d_i, h_i)_{i \in [M]})$

Fig. 4: *Σ*-protocol for statement $\{x \mid P(x)\}$. Predicate *P* is garbled by garbling scheme G.

6 CCZK: From GC to *Σ***-Protocols**

6.1 Construction

In Figure [4,](#page-17-3) we present our construction of Σ -protocol from garbling scheme $\mathsf{G} :=$ {Gb1*,* Gb2*,* Ev*,* En*,* De*,* VeC}, shuffled label commitment scheme S := {ComGb1*,* ProvEn*,* VerCom} (Appendix [B\)](#page-29-0), and NNL pseudo-random generator N := {Expand*,* SubTrees} (Appendix [A.1\)](#page-26-1).

The security is stated in Theorem [5](#page-17-2) whose proof is given in Appendix [D.4.](#page-41-0)

Theorem 5. *The protocol in Figure [4](#page-17-3) is special honest verifier zero-knowledge if garbling scheme* G *is correct and private for x satisfying* $P(x) = 1$ *, shuffled label commitment scheme* S *is hiding,* N *is pseudo-random, and hash function* Hash *is collision-resistant. It is special sound with respect to the extended statement that either* $P(x) = 1$ *or* Hash *is not collision-resistant, or* S *is not binding, or* G *is not verifiably correct.*

With the challenge space of size $\binom{M}{\tau}$, the protocol in Figure [4](#page-17-3) constitutes a proof of knowledge for the extended statement with knowledge error $\binom{M}{\tau}^{-1}$.

Performance: We estimate the communication complexity of CP-CCZK for $\{(m, r) | C(m) \wedge F(m, r)\}\$ where there are ℓ input wires of which ℓ_B are Boolean encoding and ℓ_A are arithmetic encoding. First, only small seeds s_i are needed for verifying sessions. They are compressed as nodes of the NNL expansion tree, which requires $|ST| \le \tau \cdot (\log \frac{M}{\tau}) \cdot \lambda$ assuming $|s| = |s_i| = \lambda$. The bound is obtained by considering the worst case where sessions are divided into τ groups of *M/τ* sessions each, and one evaluating session is chosen by *CH* from each group. Each group has the number of NNL nodes to open M/τ , but one session is $\log_2 \frac{M}{\tau}$. Given the node size as λ , summing it up for τ groups gives the bound.

Next we estimate evaluating sessions, where $GE = (\hat{P}_i, d_i, \tilde{x}_i, \sigma_i)_{i \in \mathsf{CH}}$ is sent. According to Equation [\(10\)](#page-32-1) in Section [B.1,](#page-29-1) the average size of (\tilde{x}_i, σ_i) from shuffled label commitment S is estimated as $|\tilde{x}| + |\sigma| \approx (\frac{|k_4|}{2} + \frac{136}{64})\ell_A + \frac{145}{64}\lambda\ell_B$. \hat{P}_i consists of \hat{C}_i and \hat{F}_i . According to Figure [1,](#page-14-0) we have $|\hat{F}_i| = |k_A| \cdot \ell_B$. Output decoding key *dⁱ* can be a hash of output wire labels of the desired output value. Since collision resistance is expected, we have $|d_i| = 2\lambda$. Summing them up to τ evaluating sessions, we have $|GE| = \tau(|\hat{C}| + |\hat{F}| + |d_i| + |\tilde{x}_i| + |\sigma_i|)$. Thus, the total proof size sent out from the prover will be

$$
|H| + |ST| + |GE| \approx 2\lambda + \tau \cdot (\log \frac{M}{\tau}) \cdot \lambda + \tau |k_A|\ell_B + \tau |\hat{C}| + 2\tau \lambda
$$

$$
+ (\frac{|k_A|}{2} + \frac{136}{64}\lambda)\tau \ell_A + \frac{145}{64}\tau \lambda \ell_B.
$$
 (5)

For $\lambda = 128$ we can select (τ, M) from, for instance, $(36, 192)$, $(30, 256)$, (22*,* 512), (18*,* 1024), (12*,* 8192) and (10*,* 32768) considering the tradeoff between communication and computation.

As a concrete example of cross-domain composition, we consider a scenario where, given a Pedersen commitment and an NTRU ciphertext, the prover proves that at least one of the committed messages fulfills the restriction represented by Boolean predicate *C*. It can be done by an OR composition of two proofs for statements St1 and St2 where

St1 := {
$$
(m,r)
$$
| $C(m) = 1 \land c_1$ = PedCom $(m;r)$ }, and
St2 := { (m,r) | $C(m) = 1 \land c_2$ = NTRUEnc $(m;r)$ }.

For 128 bits of security, *m* and *r* are 256-bits long in the Pedersen case. We estimate the case where NTRU over \mathcal{R}_q with $|q|=11$ and dimension $n=509$ (as defined in ntruhps2048509) is used for affine commitment, $\text{NTRU}_h(m;r) = hr + m$ over \mathcal{R}_q whose atomic operation is over \mathbb{Z}_q . For *m* and *r* being binary, we have input length $\ell = n$. Assuming that only *m* is input to the Boolean part, we have *ℓ*_{*A*} = $\ell_B = \ell/2$. With this setting, and $(τ, M) = (18, 1024)$ for $λ = 128$, [\(5\)](#page-18-0) is approximately $332kB$ ytes plus $18|\ddot{C}|$.

For comparison, we consider [\[37\]](#page-24-2) turned into a hybrid scheme with our garbling scheme with no further optimizations. Since [\[37\]](#page-24-2) and ours output standard *Σ*-protocols, they can compose St1 and St2 in a disjunctive manner for free using [\[26\]](#page-24-9). [\[38\]](#page-24-14) performs the disjunctive composition at the circuit level. Since the result of arithmetic equality checking is translated to Boolean for further logical composition, it is slightly more expensive than translation from Boolean to arithmetic. [\[3\]](#page-22-1) uses SNARK and is suitable for Pedersen commitments. It works only for St2 and requires CRS but composes well with a small overhead. Table [1](#page-19-1) summarizes the estimated proof sizes.

Scheme	Proof of St1	Proof of St ₂	Cost of OR	Basis
$\vert 3 \vert$	N/A	$25.5kB + CRS $	128B	$SNARG+\Sigma$
$\left 38\right $	$399kB + 46 C $	$453kB + 46 C $	910kB	MPC
37	$5.5MB + 128 GC(C) $	$68MB + 128 GC(C) $		GC.
Ours	$631kB + 18 GC(C) $	$332kB + 18 GC(C) $		GC.

Table 1: Estimated proof size for statements St1, St2, and overhead for their conjunctive composition.

6.2 Extension to Commit-and-Prove: CP-CCZK

U.Gb ₁ (1 ^λ , P): $(C, \overline{F}) \leftarrow \overline{P}$	$U.En(e, x)$: $\overline{\hat{x} \leftarrow B.En(e, x)}$	$U.De(\hat{z}, d)$: $(\hat{z}_{\mathsf{B}},\hat{z}_{\mathsf{A}}) \twoheadleftarrow \hat{z}$
$e \leftarrow B.Gb_1(1^{\lambda}, C)$	Output \hat{x} .	$(d_{\mathsf{B}}, d_{\mathsf{A}}) \leftarrow d$
Output e .		$b_{\text{B}} \leftarrow \text{B.De}(\hat{z}_{\text{B}}, d_{\text{B}})$
	U.Ev (\hat{P}, \hat{x}) :	$b_{\mathsf{A}} \leftarrow \mathsf{A}.\mathsf{De}(\hat{z}_{\mathsf{A}},d_{\mathsf{A}})$
U.Gb ₂ (P,e):	$(\hat{C}, \hat{F}) \leftarrow \hat{P}$	Output $(b_B \wedge b_A)$
$(C, F) \leftarrow P$	$\hat{z}_\text{B} \leftarrow \text{B.Ev}(\hat{C}, \hat{x})$	
$(\hat{C}, d_{\mathsf{B}}) \leftarrow \mathsf{B}.\mathsf{Gb}_2(C, e)$	$\hat{z}_A \leftarrow A.Ev(\hat{F}, \hat{x})$	
$(\hat{F}, d_{\mathsf{A}}) \leftarrow \mathsf{A}.\mathsf{Gb}_2(F, e)$	Output $\hat{z} := (\hat{z}_{\text{B}}, \hat{z}_{\text{A}}).$	
Output $\hat{P} := (\hat{C}, \hat{F})$		
and $d := (d_{\mathsf{B}}, d_{\mathsf{A}})$.		

Fig. 5: Garbling scheme U for conjunctive relation $P(x) := (C(x) \wedge F(x))$.

We extend CCZK to a commit-and-prove style by building a garbling scheme for conjunctive statement $P(x) := (C(x) \wedge F(x))$ with properties demanded in Theorem [6.](#page-20-1) The construction is illustrated in Figure [5.](#page-19-2) Recall that the resulting garbling scheme U is not fully private since it leaks information about which of B and A fails when $P(x) = 0$. It is acceptable since Theorem [6](#page-20-1) demands U being private only for *x* satisfying $P(x) = 1$ since zero-knowledge only concerns satisfying inputs by definition. It indeed is the case if both B and A are private with respect to input x, resulting in output 1. Correctness and verifiability of U are carried over from the underlying garbling schemes. Theorem [6](#page-20-1) is proved in Appendix [D.5.](#page-42-0)

Theorem 6 (Privacy-preserving Conjunctive Composition). *The garbling scheme* U *in Figure [5](#page-19-2) is correct, verifiably correct, and private with respect to x satisfying* $P(x) = 1$ *if both garbling schemes* B *and* A *are correct, verifiably correct, and private with respect to x satisfying* $C(x)$ *and* $F(x)$ *.*

Applying CCZK to the above garbling scheme U for *F* being commitment function com, we obtain commit-and-prove Σ -protocol for proving relation $C(x)$ for *x* committed with $com(x)$. That constitutes CP-CCZK.

6.3 Monotone Compositions of CP-CCZK

We consider predicates P_1, \ldots, P_n garbled by B or A, and their composition $P_0(P_1(x), \ldots, P_n(x))$ where P_0 is a monotone Boolean function. A first attempt would be to garble P_0 using the outputs of garbled P_1, \ldots, P_n as its inputs. But this does not work: in A of predicates, only output 1 is encoded with a fixed value. Any value other than the fixed encoding of 1 is translated as an encoding of 0. Thus, P cannot be evaluated when a $P_i(x)$ evaluates to zero. Therefore, P_0 must be garbled in such a way that it can be evaluated only with encodings of 1 as input. Such a *true-only* garbling is possible when P_0 is a monotone access structure where only input of 1 contributes to make the final result be 1. To garble P_0 , we assign a random value as an output of garbled P_0 and secret-share the output. Then encrypt each share with the encoded output of P_i as a key. The ciphertext is output as a part of garbled P_0 . This way, it will become clear which P_i is satisfied or not as the decryption will fail if P_i is not satisfied and an incorrect decryption key is obtained. It is however acceptable as we seek for privacy-free solution here.

For compositions with privacy, recall that garbling with privacy is needed for CCZK, which in tern produces sigma-protocols. Monotone composition of statements for Sigma protocols is a well studied topic, and we have several ways to do the task. It also applies to sequential composition of type $C \circ F$ reduced to a conjunctive composition as we observed in the previous section. Then, we show that garbling of sequential composition of type $F \circ C$ is doable.

It is required that A*.*En and B*.*De are compatible in the following sense.

Definition 12 (Out-In-Compatibility). For any d_{B} generated honestly by B.Gb *and any* \hat{y} *that* B.De $(\hat{y}, d_B) \neq \bot$, A.En $(d_B, B \cdot De(\hat{y}, d_B)) = \hat{y}$ *holds.*

This condition is satisfied with our construction of A combined with any B whose output wire labels follows the Free-Xor style. The sequential composition presented in Figure [6](#page-21-2) works for the privacy-preserving case. The following theorem is proved in Appendix [D.6.](#page-43-0)

$U.Gb(1^{\lambda}, P)$: $(\hat{C}, e, d_{\mathsf{B}}) \leftarrow \mathsf{B}.\mathsf{Gb}(1^{\lambda}, C).$ $(\hat{F}, d) \leftarrow A.Gb_2(F, d_B).$ $\hat{P} := (\hat{C}, \hat{F})$	U.Ev (\hat{P}, \hat{x}) : $(\hat{C}, \hat{F}) \leftarrow \hat{P}.$ $\hat{y} \leftarrow B.Ev(\hat{C}, \hat{x}).$ $\hat{z} \leftarrow A.Ev(\hat{F}, \hat{y}).$
Output \hat{P} , e, and d.	Output \hat{z} .
$U.En(e, x) = B.En(e, x).$	$U.De(\hat{z}, d) = A.De(\hat{z}, d).$

Fig. 6: Sequential Composition $F \circ C$.

Theorem 7 (Privacy-Preserving Sequential Composition). *If* B *is correct, decode key extractable and oblivious, and* A *is correct, verifiably correct and private with respect to y that* $f_A(y) = 1$ *, then* U *in Figure* [6](#page-21-2) *is a garbling scheme for* $P := f_A \circ f_B$ *that is correct, verifiably correct and private with respect to x that* $g(x) = 1$ *.*

7 General Composition of *Σ***-protocols**

We now present a method to compose a variety of proof systems into one following the access structure represented by a circuit. Formally, for *n* witnesses, x_1, \ldots, x_n and *n* relations, R_1, \ldots, R_n , and a monotone predicate *C* represented by a (not necessarily monotone) circuit, we construct a proof system for proving

$$
S := \{ x_{i \in [\ell]} : C(R_1(x_1), \dots, R_\ell(x_\ell)) \}
$$
\n(6)

given a proof system for each R_i . For instance, with circuit C recognizing a weighted threshold, it can be a proof of weighted threshold.

Our first step is to translate the above statement into

$$
S' := \{ (x_{i \in [\ell]}, s_{i \in [\ell]}) : C(s_1, \dots, s_\ell) \wedge_{i=1}^\ell (R_i(x_i) \vee s_i = 0) \}.
$$
 (7)

We then claim that, for monotone C , if S' is true, then so is S . Lemma [2](#page-21-1) is proved in Appendix [D.7.](#page-44-0)

Lemma 2. For any relation R_i , any monotone predicate C , and any $(x_{i \in [\ell]}, s_{i \in [\ell]}),$ *if S* ′ *is satisfied, so is S.*

It is sufficient to construct a proof system for S' . Let com_i and $open_i$ be affine commitment function and its opening verification predicate whose commitment key is implicit. We decompose *S* ′ into clauses and connect them with commitment $c_i := \text{com}_i(s_i; r_i)$ as follows:

$$
H_{R_i} := \text{PoK}\{x_i : R_i(x_i)\},
$$

\n
$$
H_{S_i} := \text{PoK}\{(s_i, r_i) : s_i = 0 \land \text{open}_i(c_i, s_i, r_i)\},
$$
 and
\n
$$
H_C := \text{PoK}\{(s_i, r_i)_{i \in [\ell]} : C(s_1, \dots, s_\ell) \land_{i \in [\ell]} \text{open}_i(c_i, s_i, r_i)\}.
$$

Consider a protocol wrapping Π_{R_i} and Π_{s_i} in a disjunctive manner as follows:

$$
\Pi_{R_i \vee s_i} := \text{PoK}\{(x_i, s_i) : R_i(x_i) \vee (s_i = 0 \wedge \text{open}_i(c_i, s_i, r_i)\}.
$$
 (8)

The final protocol $\Pi_{S'}$ for statement S' is obtained by executing Π_C and $\Pi_{R_i \vee s_i}$ for all $i \in [\ell]$ in parallel. The following claim is proved in Appendix [D.8.](#page-44-1)

Theorem 8. *The above ΠS*′ *is complete, knowledge sound, and zero-knowledge if every* com*ⁱ is hiding and binding, Π^C is zero-knowledge proof of knowledge, and every ΠRi*∨*sⁱ is a commit-and-prove witness-indistinguishable proof of knowledge.*

Suppose that a *Σ*-protocol for Π_{R_i} is given, and we construct Π_{s_i} and Π_C by CCZK obtaining corresponding *Σ*-protocols. By composing Π_{R_i} and Π_{s_i} using the disjunctive composition of [\[26\]](#page-24-9), we have a *Σ*-protocol for $\Pi_{R_i\vee s_i}$. Then final protocol $\Pi_{S'}$ that executes Π_C and $\Pi_{R_i \vee s_i}$ for all $i \in [\ell]$ in parallel as before constitute a *Σ*-protocol for *S* ′ .

Finally, we note that commitment key for each com*ⁱ* is generated by the prover. Thus, it can only be computationally hiding. Alternatively, a commitment key for statistically hiding com_i can be given as a common reference string.

Acknowledgments

The authors would like to express their sincere gratitude to Keita Xagawa for his invaluable comments on lattice-based commitments. We also deeply appreciate the detailed and constructive feedback provided by the reviewers of SCN 2024 and past conferences.

References

- 1. Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen. Non-interactive composition of sigma-protocols via share-then-hash. In *ASIACRYPT 2020*, volume 12493 of *LNCS*, pages 749–773, 2020.
- 2. Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen. Acyclicity programming for sigma-protocols. In *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 435–465, 2021.
- 3. Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zeroknowledge proofs for composite statements. In *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673, 2018.
- 4. Diego F. Aranha, Emil Madsen Bennedsen, Matteo Campanelli, Chaya Ganesh, Claudio Orlandi, and Akira Takahashi. ECLIPSE: enhanced compiling method for pedersen-committed zksnark engines. In *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 584–614, 2022.
- 5. Thomas Attema, Ignacio Cascudo, Ronald Cramer, Ivan Damgård, and Daniel Escudero. Vector commitments over rings and compressed \$\varsigma \$-protocols. In *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 173–202, 2022.
- 6. Thomas Attema and Ronald Cramer. Compressed *Σ*-protocol theory and practical application to plug & play secure algorithmics. *IACR Cryptol. ePrint Arch.*, page 152, 2020.
- 7. Thomas Attema and Ronald Cramer. Compressed \$\varsigma \$-protocol theory and practical application to plug & play secure algorithmics. In *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543, 2020.
- 8. Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 65–91, 2021.
- 9. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed \$\varsigma \$ protocol theory for lattices. In *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, 2021.
- 10. Thomas Attema, Serge Fehr, and Nicolas Resch. Generalized special-sound interactive proofs and their knowledge soundness. In *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 424–454. Springer, 2023.
- 11. Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, and Ivan Visconti. Efficient proofs of knowledge for threshold relations. In *ESORICS 2022, Part III*, volume 13556 of *LNCS*, pages 42–62, 2022.
- 12. Marshall Ball, Tal Malkin, and Mike Rosulek. Garbling gadgets for boolean and arithmetic circuits. In *ACM CCS 2016*, pages 565–577, 2016.
- 13. Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and z2k. In *ACM CCS 2021*, pages 192–211, 2021.
- 14. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM CCS 2012*, pages 784–796, 2012.
- 15. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518, 2019.
- 16. David Butler, Andreas Lochbihler, David Aspinall, and Adrià Gascón. Formalising \$\varsigma\$-protocols and commitment schemes using crypthol. *J. Autom. Reason.*, 65(4):521–567, 2021.
- 17. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33, 2021.
- 18. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In *ACM CCS 2019*, pages 2075–2092, 2019.
- 19. Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530, 2016.
- 20. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58, 2015.
- 21. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 112–141, 2016.
- 22. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 63–92, 2016.
- 23. Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, 1997.
- 24. Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. $\mathrm{SPDZ}_{\mathrm{g}\mathrm{k}}\colon$ Efficient MPC mod 2^{k} for dishonest majority. In *CRYPTO 2018, Part* 2 *II*, volume 10992 of *LNCS*, pages 769–798, 2018.
- 25. Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 2000*, volume 1751 of *LNCS*, pages 354–372, 2000.
- 26. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO 1994*, volume 839 of *LNCS*, pages 174–187, 1994.
- 27. Hongrui Cui and Kaiyi Zhang. A simple post-quantum non-interactive zeroknowledge proof from garbled circuits. In *Inscrypt 2021*, volume 13007 of *LNCS*, pages 269–280, 2021.
- 28. Ivan Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 17–27, 1989.
- 29. Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 823–852, 2020.
- 30. Pierre-Alain Fouque, Adela Georgescu, Chen Qian, Adeline Roux-Langlois, and Weiqiang Wen. A generic transform from multi-round interactive proof to NIZK. In *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 461–481, 2023.
- 31. Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 191–219, 2015.
- 32. Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptol. ePrint Arch.*, 2020:315, 2020.
- 33. Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. In *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 499–529, 2018.
- 34. François Garillot, Yashvanth Kondi, Payman Mohassel, and Valeria Nikolaenko. Threshold schnorr with stateless deterministic signing from standard assumptions. In *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 127–156, 2021.
- 35. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, 2013.
- 36. Aarushi Goel, Mathias Hall-Andersen, Gabriel Kaptchuk, and Nicholas Spooner. Speed-stacking: Fast sublinear zero-knowledge proofs for disjunctions. In *EURO-CRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 347–378, 2023.
- 37. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 397–429, 2016.
- 38. David Heath, Vladimir Kolesnikov, and Jiahui Lu. Efficient generic arithmetic for KKW - practical linear mpc-in-the-head NIZK on commodity hardware without trusted setup. In *CSCML 2021*, volume 12716 of *LNCS*, pages 414–431, 2021.
- 39. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *ACM STOC 2007*, pages 21–30, 2007.
- 40. Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 2013*, pages 955–966, 2013.
- 41. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pages 525–537, 2018.
- 42. Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 286–313, 2018.
- 43. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP 2008, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *LNCS*, pages 486–498, 2008.
- 44. Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO 1987*, volume 293 of *LNCS*, pages 369–378, 1987.
- 45. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62, 2001.
- 46. Ky Nguyen, Miguel Ambrona, and Masayuki Abe. WI is almost enough: Contingent payment all over again. In *ACM CCS 2020*, pages 641–656, 2020.
- 47. É. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.
- 48. Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In *USENIX Security 2021*, pages 501–518, 2021.
- 49. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *IEEE FOCS 1982*, pages 160–164, 1982.
- 50. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *IEEE FOCS 1986*, pages 162–167, 1986.
- 51. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250, 2015.

Appendix

A Other Building Blocks

A.1 NNL Pseudo-random Generator

An NNL pseudo-random generator is a pair of polynomial-time algorithms (Expand*,* SubTrees) that:

- Expand $(RT) \rightarrow (r_i)_i$: Given $RT = (s, \ell) \in \{0, 1\}^{\lambda} \times \mathbb{N}$, it outputs $(r_i)_{i \in [\ell]} \in$ $\{0,1\}^{\lambda \times \ell}$. For notational convenience, we overlay this function to its *k*-fold repetition that takes $RT = (RT_1, \ldots, RT_k)$ as input, executes $\textsf{Expand}(RT_i)$ for $i = 1, \ldots, k$, and outputs the results concatenated.
- SubTrees $(RT, I) \rightarrow ST$: Given $RT = (s, \ell) \in \{0, 1\}^{\lambda} \times \mathbb{N}$, and indices $I \subseteq [\ell]$, it outputs $ST = (s_j, \ell_j)_{j \in [k]}$ that, for $(r'_i)_{i \in I} = \mathsf{Expand}(ST)$ and $(r_i)_{i \in [\ell]} \leftarrow \textsf{Expand}(RT), r'_i = r_i \text{ hold for all } i \in I.$

Expand forms a binary tree whose root node value is *s*, and child nodes are generated by applying a length-doubling pseudo-random generator to the parent node. The standard indexing identifies every node for binary trees. For ease of notation, we assume that every node value is accompanied by the index implicitly. Therefore, Expand consistently expands sub-trees from intermediate nodes given by SubTrees to the whole tree. The convention about indexing also applies to functions and variables in Merkle trees.

Definition 13. *An NNL pseudo-random generator is* pseudo-random *if, for all polynomial ℓ and all polynomial-time adversaries* A*, the following advantage is upper bounded by a negligible function in λ:*

$$
\Pr\begin{bmatrix} (I,st) \leftarrow \mathcal{A}(1^{\lambda},\ell) \\ s \leftarrow \{0,1\}^{\lambda} \\ RT := (s,\ell) \\ (r_i)_{i\in [\ell]} \leftarrow \textnormal{Expand}(RT) \\ ST \leftarrow \textnormal{SubTree}(RT,I) \\ b \leftarrow \mathcal{A}(st,(r_i)_{i\in [\ell]\backslash I}, ST) \end{bmatrix} - \Pr\begin{bmatrix} (I,st) \leftarrow \mathcal{A}(1^{\lambda},\ell) \\ s \leftarrow \{0,1\}^{\lambda} \\ RT := (s,\ell) \\ (r_i)_{i\in [\ell]} \leftarrow \{0,1\}^{\lambda\times \ell} \\ ST \leftarrow \textnormal{SubTree}(RT,I) \\ \delta T \leftarrow \textnormal{SubTree}(RT,I) \\ b \leftarrow \mathcal{A}(st,(r_i)_{i\in [\ell]\backslash I}, ST) \end{bmatrix}
$$

A.2 Merkle Tree

We consider a family of Merkle trees $\mathcal{MT} = {\mathcal{MT}_{\lambda}}$ parameterized by security parameter λ . MT $\in \mathcal{MT}_{\lambda}$ is a tuple of polynomial-time deterministic algorithms (Com*,* Proof*,* Ver) that:

- $\textsf{Com}(L) \to R$: Given leaf-values $L := (L_1, \ldots, L_\ell) \in \{0,1\}^{\lambda \times \ell}$ as input, it outputs a root value *R*.
- Proof $(L, I) \rightarrow \Pi$: Given leaf-values *L* and a set of indices $I := (i_1, \ldots, i_k)$ as input, it outputs Π , siblings for every path from leaf i_j to the root.

- $\text{Ver}(S, \Pi) \rightarrow R$: Given a subset of indexed leaf-values *S* and siblings Π as input, it computes the root value *R*.

It is correct if, for all $L \in \{0,1\}^{\lambda \times \ell}$, $I \subseteq [\ell], S \subseteq L, R \leftarrow \text{Com}(L), \Pi \leftarrow$ Proof (L, I) , it holds that $R = \text{Ver}(S, \Pi)$.

A Merkle tree is unforgeable if an unauthorized leaf passes the verification only with negligible probability.

Definition 14 (Unforgeability). *A Merkle tree family* MT *is unforgeable if, for any polynomial-time adversary* A*, any polynomial ℓ, the following probability is negligible in λ:*

$$
\Pr\left[\begin{matrix}\mathsf{MT} \leftarrow \mathcal{MT}_{\lambda} \\ (L, S, \Pi) \leftarrow \mathcal{A}(1^{\lambda}) \end{matrix} : \begin{matrix} L \in \{0, 1\}^{\lambda \times \ell} \land S \nsubseteq L \land \\ \mathsf{MT}.\mathsf{Com}(L) = \mathsf{MT}.\mathsf{Ver}(S, \Pi) \end{matrix}\right]\right]
$$

We require a Merkle tree not to leak information about the unopened, correlated leaves even if a half of the leaves are opened. We call this property as MT correlation robustness formally defined as follows.

Definition 15 (MT Correlation Robustness). For relation $R \in \mathcal{R}$, let $\mathcal{D}_R = \{L_1, \ldots, L_{2\ell} \, | \, \forall i \in [\ell], \, R(L_i, L_{i+\ell}) = 1\}.$ A Merkle tree MT is MT corre*lation robust with respect to* R*, if, for any polynomial-time adversary* A*, any* $(\xi_1, \ldots, \xi_\ell) \in \{0,1\}^\ell$, and $I := (i + \xi_i \cdot \ell)_{i \in [\ell]}$, the following advantage is negligible *in λ:*

$$
\Pr\left[\begin{matrix} R \leftarrow \mathcal{R}, L \leftarrow \mathcal{D}_R \\ h \leftarrow \text{Com}(L) \\ \pi \leftarrow \text{Proof}(L, I) \\ b \leftarrow \mathcal{A}(h, \pi, L_{i \in I}) \end{matrix}\right] \right] - \Pr\left[\begin{matrix} R \leftarrow \mathcal{R}, L \leftarrow \mathcal{D}_R \\ I' := [2\ell] \setminus I \\ \forall i \in I, L'_i := L_i \\ \forall i \in I', L'_i \leftarrow \{0, 1\}^{\lambda} \\ L' := \{L'_i \mid i \in [2\ell]\} \\ h \leftarrow \text{Com}(L') \\ \pi \leftarrow \text{Proof}(L', I) \\ b \leftarrow \mathcal{A}(h, \pi, L_{i \in I}) \end{matrix}\right]\right]
$$

A.3 Key Derivation Function

A key derivation function is indistinguishable from random if its output on random input is indistinguishable from random.

Definition 16 (KDF Indistinguishability). *Let* KDF *be a key derivation function* $KDF : \mathcal{D}(\lambda) \to \{0,1\}^{\lambda}$. Let \mathcal{O}_{KDF} be an oracle that, given a query, it *samples* $x \leftarrow_R \mathcal{D}(\lambda)$ *and returns* KDF(*x*)*. Let* \mathcal{O}_R *denote an oracle that, given a query, it returns a uniformly chosen value from* $\{0,1\}^{\lambda}$. KDF *is indistinguishable from random if, for all polynomial-time adversary* \mathcal{A} *, oracles* $\mathcal{O}_{\mathsf{KDF}}$ *and* \mathcal{O}_R *are indistinguishable.*

We follow the notion of *correlation robustness* [\[34\]](#page-24-6) of KDF with minor adjustment for the privacy setting. Roughly, it states that, it is hard for the adversary to distinguish oracle $\mathcal{O}_{\mathsf{KDF}}$ that, on receiving (i, k_i) , returns $\mathsf{KDF}(i, k_i \oplus \Delta)$ for random fixed Δ , and oracle \mathcal{O}_R that evaluates a random function.

 $\textbf{Definition 17 (Correlation Robustness [34])}.$ $\textbf{Definition 17 (Correlation Robustness [34])}.$ $\textbf{Definition 17 (Correlation Robustness [34])}.$ *Let* KDF : $\{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \rightarrow$ \mathcal{R} *be a key derivation function, and* \mathcal{U} *be all functions over* $\{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \mathcal{R}$ *. Let* \mathcal{O}_{Δ} *and* \mathcal{O}_{R} *be oracles that* $\mathcal{O}_{\Delta}(i,x) := \mathsf{KDF}(i,x \oplus \Delta)$ *and* $\mathcal{O}_{\mathsf{R}}(i,x) :=$ R(*i, x*)*. Oracles accept queries with unique i.* KDF *is correlation robust if, for all polynomial-time algorithms* A*, advantage function*

$$
\Pr_{\Delta \in \{0,1\}^{\lambda-1} \parallel 1} \left[\mathcal{A}^{\mathcal{O}_{\Delta}}(1^{\lambda}) = 1 \right] - \Pr_{\mathsf{R} \in \mathcal{U}} \left[\mathcal{A}^{\mathcal{O}_{\mathsf{R}}}(1^{\lambda}) = 1 \right] \tag{9}
$$

is negligible in λ.

For privacy-free setting, Δ can be chosen from $\{0,1\}^{\lambda}$.

We note that the correlation robustness in the above definition is implied by circular correlation robustness introduced in [\[51\]](#page-25-9) whose oracles are $\mathcal{O}_{\Delta}(i, x, b)$:= $KDF(i, x \oplus \Delta) \oplus (b \cdot \Delta)$ and $\mathcal{O}_R(i, x, b) := R(i, x, b)$. This observation is important because we use common label generation mechanism for Boolean that requires circular correlation robustness and arithmetic garbling that require correlation robustness. By the above implication, our construction eventually rely on the circular correlation robustness.

A.4 Symmetric-key Encryption

A symmetric-key encryption scheme SE consists of a key generation algorithm Gen, encryption algorithm Enc and decryption algorithm Dec. We assume that Gen uniformly samples a key from $\{0,1\}^{\lambda}$, and follow the standard notion of indistinguishability against chosen plaintext attacks. By M and K , we denote the message and key spaces defined implicitly for security parameter λ . Indistinguishability against chosen message attacks (IND-CPA) is defined in the standard manner where a challenge ciphertext of either of the two messages chosen by the adversary is indistinguishable in the presence of the encryption oracle.

A.5 Secret Sharing Scheme

Secret sharing scheme SS := (Share*,* Rec*,* Ver) is a tuple of polynomial-time algorithms that

- $t \leftarrow$ Share(Γ, w) is a sharing algorithm that takes a description of access structure *Γ* over $(1, \ldots, n)$, and a secret *w*, and outputs shares $t = (t_1, \ldots, t_n)$,
- $w \leftarrow \text{Rec}(T, t')$ is a reconstruction algorithm that takes $t' = (t_{i_1}, \ldots, t_{i_{n'}})$ and outputs *w* or ⊥, and
- $0/1 \leftarrow \text{Ver}(\Gamma, t)$ is a verification algorithm that takes a full set of shares $t = (t_1, \ldots, t_n)$ and outputs 1 or 0.

It is correct if, for any *w* in a fixed domain, $t \leftarrow \textsf{Share}(r, w)$, $T \in \Gamma$, $U \notin \Gamma$, $w = \text{Rec}(F, (t_i)_{i \in T})$ and $w \neq \text{Rec}(F, (t_i)_{i \in U})$.

It is secure if, for any polynomial-time adversary A , any $U \notin \Gamma$, and uniformly chosen *w*, $w = A((t_i)_{i \in U})$ happens only with negligible probability.

A secret sharing scheme is verifiable if $\text{Ver}(\Gamma, t)$ outputs 1 if and only if $t \in$ Share(*Γ, w*) for some *w*. Shamir's secret sharing is verifiable for threshold structure *Γ* with a threshold of one-third. To accept more complex access structures, we can use a verifiable secret sharing scheme where Share outputs a public parameter that is given as input to Rec and Ver.

B Shuffled Label Commitment

B.1 Definitions

The role of the shuffle label commitment scheme is threefold: 1) to commit to correlated pairs of labels for a garbled circuit in a pairwise shuffled manner, 2) to open one of each paired label, and 3) to open all labels in the correct order.

Definition 18. *A shuffled label commitment scheme* S *for garbling scheme* B *is a tuple of polynomial-time algorithms* (ComGb1*,* ProvEn*,* VerCom) *that:*

 $\textsf{ComGb}_1(1^\lambda, P; s) \to (e, h)$: Given security parameter 1^λ and predicate P as *input, it generates encoding key e and commits to e by h.*

 $\mathsf{ProvEn}(P, x, s) \to (\tilde{x}, \sigma)$ *: Given predicate P, input value x, and coins s (used*) in ComGb₁) as input, it outputs compressed encoded input \tilde{x} and commitment *verification information σ.*

 $VerCom(\tilde{x}, \sigma) \rightarrow h$ *: Given compressed encoded input* \tilde{x} *and commitment verification information* σ *as input, it reconstructs commitment* h *.*

It is accompanied by supplemental functions, (CompEn*,* CompEn[−]¹ *,*UnComp) *that:*

CompEn(P, s, x) $\rightarrow \tilde{x}$ *: Given predicate* P *, seed s, and value x, it outputs compressed encoding* \tilde{x} *of* x *.*

Comp $En^{-1}(P, s, \tilde{x})$ → *x:* Given P, s and compressed encoding \tilde{x} *as input, it outputs x.*

UnComp $(P, \tilde{x}) \rightarrow \hat{x}$ *: Given P* and compressed \tilde{x} *, it outputs standard* \hat{x} *.*

It is correct if, for all $s \in \{0,1\}^*$, predicate P, x in the domain of P, $(e, h) \leftarrow$ $\textsf{ComGb}_1(1^\lambda, P; s), (\tilde{x}, \sigma) \leftarrow \textsf{ProvEn}(P, x, s), e' \leftarrow \textsf{Gb}_1(1^\lambda, P; s), \text{ and } \tilde{x}' \leftarrow \textsf{CompEn}$ (P, s, x) , it holds that $e = e'$, $\tilde{x} = \tilde{x}'$, $h = \text{VerCom}(\tilde{x}, \sigma)$, $x = \text{CompEn}^{-1}$ $(P, s, \textsf{CompEn}(P, s, x))$, and $\textsf{UnComp}(P, \textsf{CompEn}(P, s, x)) = \textsf{En}(e, x)$.

Security properties are *binding* and *hiding*. Intuitively, binding property states that successful opening of commitment h with encoded input \hat{x} implies extraction of embedded *x*.

Definition 19 (Binding). *A shuffled label commitment scheme* S *for garbling scheme* B *is binding if, for any polynomial-time algorithm A, the following probability is negligible in λ:*

$$
\Pr\left[\begin{matrix}(e,h) \leftarrow \mathsf{ComGb}_1(1^\lambda,P;s) & h = \mathsf{VerCom}(\tilde{x},\sigma) \wedge \\ (\tilde{x},\sigma) \leftarrow A(s) & \perp = \mathsf{CompEn}^{-1}(P,s,\tilde{x})\end{matrix}\right]\right]
$$

Hiding property is that correct opening (\tilde{x}, σ) of commitment *h* does not leak information about e and x more than \tilde{x} does. Combined with Lemma [1,](#page-10-2) it implies that no information about *x* is leaked.

Definition 20 (Hiding). *A shuffled label commitment scheme* S *for garbling scheme* B *is* hiding *if there exists a polynomial-time simulator* S_S *that, for all predicate P and x in the domain of P, and all polynomial-time adversaries* A*, the following advantage is negligible in* λ *:*

$$
\Pr\begin{bmatrix}(e,h) \leftarrow \mathsf{ComGb}_1(1^\lambda,P;s)\\ (\tilde{x},\sigma) \leftarrow \mathsf{ProvEn}(P,x,s)\\ b \leftarrow \mathcal{A}(\tilde{x},\sigma,h)\end{bmatrix}\;:\; b=1\begin{bmatrix} e \leftarrow \mathsf{Gb}_1(1^\lambda,P;s)\\ \tilde{x} \leftarrow \mathsf{CompEn}(P,s,x)\\ (h,\sigma) \leftarrow \mathcal{S}_\mathsf{S}(P,\tilde{x})\\ b \leftarrow \mathcal{A}(\tilde{x},\sigma,h)\end{bmatrix}\;:\; b=1\end{bmatrix}
$$

B.2 Construction

We construct a shuffled label commitment scheme, S, from NNL pseudo-random generator N (Appendix [A.1\)](#page-26-1), Merkle tree scheme MT (Appendix [A.2\)](#page-26-2), and key derivation function $KDF: \{0,1\}^* \to \mathcal{R}$ (Appendix [A.3\)](#page-27-0). We begin by building functions, GbB and GbA that, given two random coins, *r* and *p*, generate a pair of labels for *i*-th Boolean or arithmetic wire. Let *∆* be a global parameter sampled as shown in Figure [1.](#page-14-0)

We then show an algorithm, GetLeaf, that generates label pairs for *P* using GbB and GbA with random coins generated by NNL pseudo-random generator, and associate the labels to the leaves of Merkle tree in an appropriate place and form. Each label pairs $(k_i^{b_i}, k_i^{\bar{b}_i})$ permuted randomly is associated to *i*-th and $i + \ell$ -th leaves, L_i and $L_{i+\ell}$. However, directly putting labels as a leaf value causes a problem when labels are taken from a small domain, as in the case of arithmetic labels. Our idea is to generate $k_i^{b_i}$ directly from random coins r_i and assign_{*r*}^{*i*}, which has more entropy than $k_i^{b_i}$, to leaf L_i . For $i + \ell$ -th leaves, however, $k_i^{\bar{b}_i}$ depends on $k_i^{b_i}$ and cannot be generated randomly. Hence, we pad random coins p_i to gain entropy.

We are ready to present the main functions of S.

 $\mathsf{ComGb}_1(1^\lambda,P;s)$ $(L, \pi, e) \leftarrow$ GetLeaf(*P*, *s*) $h \leftarrow \text{MT}$ *.* Com (L) Output (*e, h*). VerCom(˜*x, σ*) $(ST_1, K_2, I_1, I_2) \leftarrow \tilde{x}$ $(k_i)_{i \in I_2} \leftarrow K_2$ $(\Pi, ST_{2A}) \leftarrow \sigma$ $I_{2A} := \{ i \in I_2 \, | \, i - \ell \in A$ -Labels} $(L_i)_{i \in I_1} \leftarrow \mathsf{N}$ *.* Expand (ST_1) $(p_i)_{i \in I_{2A}}$ ← N.Expand(ST_{2A}) $∀i ∈ I₂$ If *i* − *ℓ* ∈ A-Labels $L_i := k_i || p_i$ If *i* − *ℓ* ∈ B-Labels $L_i := k_i$ $S := (L_i)_{i \in I_1 \cup I_2}$ $h \leftarrow \textsf{MT.Ver}(S, \Pi)$ Output *h*. ProvEn(*P, x, s*) $\ell \leftarrow$ InputLen(*P*) $(L, \pi, e) \leftarrow$ GetLeaf (P, s) $I := \{i + \ell(x_i \oplus \pi_i) \mid i \in [\ell]\}$ $I_1 := \{i \in I \mid i \leq \ell\}, I_2 := I \setminus I_1$ $I_{2A} := \{ i \in I_2 \, | \, i - \ell \in A$ -Labels} $RT := (s, 2)$ $(s_1, s_2) \leftarrow \mathsf{N}$. Expand (RT) $RT_1 := (s_1, \ell)$ $ST_1 \leftarrow \mathsf{N}.\mathsf{SubTrees}(RT_1,I_1)$ $K_2 := \{ k_{i-\ell}^{x_{i-\ell}} \in e \mid i \in I_2 \}$ \tilde{x} := (*ST*₁*, K*₂*, I*₁*, I*₂*)* $$ $RT_2 := (s_2, \ell)$ $ST_{2A} \leftarrow \mathsf{N}.\mathsf{SubTrees}(RT_2, I_{2A})$ $\sigma := (\Pi, ST_{2A})$ Output (\tilde{x}, σ) .

Accompanied functions for handling compressed encoding are given below.

CompEn(*P, s, x*): $\ell \leftarrow$ InputLen(*P*) $(L, \pi, e) \leftarrow$ GetLeaf (P, s) $I_1 := \{i \in [\ell] \mid x_i \oplus \pi_i = 0\}$ $I_2 := \{i + \ell \, | \, i \in [\ell], x_i \oplus \pi_i = 1\}$ Output *x*. $(s_1, s_2) \leftarrow \mathsf{N}$. Expand $(s, 2)$ $RT_1 := (s_1, \ell)$ $ST_1 \leftarrow \mathsf{N}.$ SubTrees (RT_1, I_1) $K_2 := \{ k_{i-\ell}^{x_{i-\ell}} \in e \mid i \in I_2 \}$ \tilde{x} := (*ST*₁*, K*₂*, I*₁*, I*₂*)* Output \tilde{x} . CompEn^{-1} (P, s, \tilde{x}) : $(L, \pi, e) \leftarrow$ GetLeaf (P, s) $\ell \leftarrow$ InputLen (P) $\hat{x} \leftarrow$ UnComp (P, \tilde{x}) x ← En⁻¹(e, \hat{x}) UnComp (P, \tilde{x}) : $(ST_1, K_2, I_1, I_2) \leftarrow \tilde{x}$ $(r_i)_{i \in I_1}$ ← N.Expand(ST_1) ∀ *i* ∈ B-Labels $k'_i := r_i$ ∀ *i* ∈ A-Labels $k'_i \leftarrow \mathsf{KDF}(r_i)$ $(k'_{i-\ell})_{i \in I_2} \leftarrow K_2$ \hat{x} := $(k'_{i})_{i \in [\ell]}$ Output *x*ˆ.

Whenever \hat{x} in the compressed form is parsed into (ST_1, K_2, I_1, I_2) , its proper formatting is implicitly verified. Namely, it checks if *ST*¹ follows the form of nodes of a NNL tree, K_2 is a list of labels, $I_1 \subseteq [\ell], I_2 \subseteq [\ell+1, 2\ell],$ and for $I'_2 := \{i - \ell \mid i \in I_2\}, I_1 \cup I'_2 = [\ell]$. If the check fails, the algorithm halts with output ⊥.

B.3 Performance

We estimate the size of (\tilde{x}, σ) , which is the most relevant to the construction in Section [6.1.](#page-17-1) Consider the minimal case of $\ell = 2$ where labels (k_1^0, k_1^1) and (k_2^0, k_2^1) are allocated to leaves (L_1, L_3) and (L_2, L_4) pairwise shuffled. If L_1 is opened, the other side of the pair, *L*3, is kept hidden. Namely, a hash value of *L*³ is published. If *L*³ is opened, *L*¹ has to be kept hidden. The same for the other pair. To open L_1 and L_2 at the same time, a corresponding NNL node is published. Then, a corresponding MT node is published to hide L_3 and L_4 altogether. Unfortunately, it is not symmetric when opening and hiding are altered. To open L_3 and L_4 , corresponding labels must be published one by one because they are not generated directly from the NNL tree. When *L*³ and *L*⁴ are arithmetic labels, random padding parts can be taken from the NNL tree. This process is done in ProvEn.

Extending the above to $\ell = 4$ and counting all 16 possible opening and hiding patterns of leaves with an assumption that NNL nodes and labels for Boolean wires are λ bits, and MT nodes are 2λ bits, the average size of (\tilde{x}, σ) produced by ProvEn is estimated as $(|k_A|/2 + 136\lambda/64)$ *l* when all labels are for arithmetic with size $|k_A|$, and $(145\lambda/64)\ell$ when all labels are for Boolean. For $\ell = \ell_A + \ell_B$ where ℓ_A and ℓ_B are a number of input wires of *P* for arithmetic and Boolean, respectively, we have

$$
|\tilde{x}| + |\sigma| \approx \left(\frac{|k_A|}{2} + \frac{136}{64}\lambda\right)\ell_A + \frac{145}{64}\lambda\ell_B.
$$
 (10)

The naive construction with individual commit and open approach, $((|k_A| + \lambda) +$ 2λ) $\ell_A + 4\lambda\ell_B$ in the same setting. For $|k_A| = \lambda$ and $\ell_A = \ell_B = \ell/2$ for instance, our construction costs $313/128 \cdot \lambda \ell$ whereas that of naive approach is $4\lambda \ell$. Thus, our shuffled label commitment with compressed encoding saves $\approx 39\%$ over the naive approach.

B.4 Security

Theorem 9. *Shuffled label commitment scheme* S *in Section [B.2](#page-30-0) is binding if Merkle tree* MT *is unforgeable.*

Proof. Suppose that there is an adversary that, given 1^{λ} and *P*, outputs (s, \tilde{x}, σ) that satisfies $h = \text{VerCom}(\tilde{x}, \sigma)$ for $(e, h) \leftarrow \text{ComGb}_1(1^\lambda, P; s)$, but fails to recover embedded value, $\bot = \textsf{CompEn}^{-1}(P, s, \tilde{x})$.

Recall that ComGb_1 executes $(L, \pi, e) \leftarrow \mathsf{GetLeaf}(P, s)$ and $h \leftarrow \mathsf{MT}.\mathsf{Com}(L)$. Since $\textsf{CompEn}^{-1}(P, s, \tilde{x})$ outputs \perp , we have $\perp \leftarrow \textsf{En}^{-1}(e, \hat{x})$ for $\hat{x} \leftarrow \textsf{UnComp}(P, \tilde{x})$. Consider the case that $\hat{x} = \perp$. It can only happen when parsing (RT_1, K_2, I_1, I_2) \tilde{x} in UnComp(P, \tilde{x}) fails. Since the same parsing is done in VerCom, it ends up with $\bot \leftarrow \mathsf{VerCom}(\tilde{x}, \sigma)$ contradicting that it outputs *h*. On the contrary, consider the case that $\hat{x} \neq \bot$. En⁻¹(*e,* \hat{x}) outputs \bot only if \hat{x} is not a subset of *e*. (More precisely, there is an element, k_i in \hat{x} that does not match either k_i^0 or k_i^1 in *e*.) Denote this by $\hat{x} \not\subset e$. Recall that $\mathsf{VerCom}(\tilde{x}, \sigma)$ computes *S* that determines \hat{x} , and *Π* that satisfies $h = \text{MT}.\text{Ver}(S, \Pi)$ for $h \leftarrow \text{MT}.\text{Com}(L)$ computed in ComGb₁($1^{\lambda}, P$; *s*). Since *L* also uniquely determines *e* in the same manner, if *S* ⊂ *L*, then \hat{x} ⊂ *e*. However, since we have \hat{x} ⊄ *e*, we conclude that S ⊄ *L*. Thus, A producing such (s, \hat{x}, σ) , we can compute (L, S, Π) breaks unforgeability of MT. $□$

We consider relations between *i*-th and $i + \ell$ -th leave values corresponding to arithmetic and Boolean wires. Let *R^A* and *R^B* be relations for arithmetic and Boolean labels, respectively, as defined below. *R^A* is a fixed relation (for any fixed δ), and R_B is parameterized by Δ .

$$
\frac{R_A(L_i, L_{i+\ell}) :}{k'_i || p_i \leftarrow L_{i+\ell}} \n\frac{R_B(L_i, L_{i+\ell}) :}{\text{Output } L_{i+\ell} \stackrel{\text{'}}{=} L_i \oplus \Delta}
$$
\n
$$
\pi_i := \text{LSB}(L_i) \oplus \text{LSB}(p_i)
$$
\n
$$
k_i := \text{KDF}(i, L_i)
$$
\n
$$
\text{Output } k'_i \stackrel{\text{'}}{=} k_i + (1 - 2\pi_i)\delta
$$

Theorem 10. *Shuffled label commitment scheme* S *in Section [B.2](#page-30-0) is hiding if* N *is pseudo-random and* MT *is MT correlation robust for correlations R^B and RA.*

Proof. We construct S_S as follows.

 $\mathcal{S}_\mathsf{S}(P,\tilde{x})$ 01: ℓ ← InputLen(*P*) 02: (ST_1, K_2, I_1, I_2) ← $\tilde{x}, k_{i \in I_2}$ ← K_2 03: $I'_1 := \{i + \ell \mid i \in I_1\}$ 04: $I'_2 := \{i - \ell \mid i \in I_2\}$ 05: $(r_i)_{i \in I_1}$ ← N.Expand (ST_1) 06: $\forall i \in I'_2, r_i \leftarrow \{0, 1\}^{\lambda}$ 07: $s_2 \leftarrow \{0,1\}^{\lambda}$ 08: $RT_2 := (s_2, \ell)$ 09: (*pi*)*i*∈[*ℓ*+1*,*2*ℓ*] ← N*.*Expand(*RT*2) 10: $\forall i \in I'_1,$ 11: if *i* − *ℓ* ∈ B-Labels 12: $r_i \leftarrow \{0, 1\}^{\lambda}$ 13: if *i* − *ℓ* ∈ A-Labels 14: $r_i \leftarrow \mathcal{R} \times \{0,1\}^{\lambda}$ 15: $\forall i \in I_2$ 16: if *i* − *ℓ* ∈ B-Labels 17: $r_i = k_i$ 18: if $i - \ell \in A$ -Labels 19: $r_i := k_i || p_i$ 20: $L := (r_1, \cdots, r_{2\ell})$ 21: $h \leftarrow \text{MT}.\text{Com}(L)$ 22: $\Pi \leftarrow \text{MT}.Proof(L, I_1 \cup I_2)$ 23: $I_{2A} := \{i \in I_2 \, | \, i - \ell \in \mathsf{A}\text{-Labels}\}\$ 24: $ST_{2A} \leftarrow \mathsf{N}.SubTrees(RT_2, I_{2A})$ 25: $\sigma := (\Pi, ST_{2A})$ 26: Output (h, σ)

Let H_0 be the game with simulator S_S in the right-hand term of the advantage defined in Definition [20.](#page-30-1)

 H_1 : We replace random choices of r_i in step 12 and 14 with the following process. Δ is the one used in Gb_1 .

12:
$$
r_i := r_{i-\ell} \oplus \Delta
$$

\n14-1: $\pi_i := \text{LSB}(r_{i-\ell}) \oplus \text{LSB}(p_i)$
\n14-2: $k_i \leftarrow \text{KDF}_{\mathcal{R}}(r_{i-\ell}) + (1 - 2\pi)\delta$
\n14-3: $r_i := k_i || p_i$

This makes r_i and $r_{i+\ell}$ for $i \in I_1$ in relation R_B and R_A . The difference caused by this change is negligible due to the MT correlation robustness of MT for correlations *R^B* and *RA*.

*H*₂*:* Let *s* be the random coins used in Gb_1 . We replace step 05 with $(s_1, s_2) \leftarrow$ N.Expand(*s*, 2) and $(r_i)_{i \in [\ell]}$ ← N.Expand(s_1, ℓ). This transition is a cosmetic change. It is because that r_i for $i \in I_1$ is generated in the same way using the same seed *s* in Gb_1 . Regarding r_i , r_i for $i \in I'_2$ are overwritten in step 06 as before. Hence, Game H_2 is equivalent to H_1 .

*H*₃: We remove step 06. It means that the random choice of r_i for $i \in I'_2$ is replaced with the ones generated by N*.*Expand in step 05. Due to the pseudorandomness of NNL, Game H_3 is indistinguishable from Game H_2 .

 H_4 : We remove step 07. It means that the random choice of s_2 is replaced with the one generated by N*.*Expand in step 05. Due to the pseudo-randomness of NNL, Game H_4 is indistinguishable from Game H_3 .

The process done in H_4 is exactly what is done in the real game in the left-hand term of the advantage defined in Definition [20.](#page-30-1) ⊓⊔

What remains is to make it clear under what assumption such MT correlation robust MT can be constructed. For two variables L_i and $L_{i+\ell}$ in relation $R_A(L_i, L_{i+\ell}) = 1$, we consider sampling one given the other. For this purpose, we define sets, D_1 , that defines all possible choices of $L_{i+\ell}$ when L_i is fixed. D_2 is defined as the set of possible L_i when $L_{i+\ell}$ is fixed.

$$
D_1(L_i) := \left\{ L_{i+\ell} \mid \begin{aligned} p &\in \{0,1\}^{\lambda} \\ \pi & = \text{LSB}(p) \oplus \text{LSB}(L) \\ L_{i+\ell} &= (\text{KDF}(i, L_i) + (1 - 2\pi)\delta) || p \end{aligned} \right\},
$$
\n
$$
D_2(L_{i+\ell}) := \left\{ L_i \mid \begin{aligned} L_i &\in \{0,1\}^{\lambda} \\ k || p &\leftarrow L_{i+\ell} \\ \pi &= \text{LSB}(p) \oplus \text{LSB}(L_i) \\ k &= \text{KDF}(i, L_i) + (1 - 2\pi)\delta \end{aligned} \right\}
$$

We then define oracles as follows.

$$
\frac{\mathcal{O}_{A1}(i, L_i):}{L_{i+\ell} \leftarrow_R D_1(L_i)} \qquad \qquad \frac{\mathcal{O}_{A2}(i+\ell, L_{i+\ell}):}{L_i \leftarrow_R D_1(L_{i+\ell})} \qquad \qquad \frac{\mathcal{O}_B(i, L_i):}{\text{Return Hash}(L_i \oplus \Delta)}.
$$
\n
$$
\text{Return Hash}(L_{i+\ell}). \qquad \qquad \text{Return Hash}(L_i)
$$

Let \mathcal{O}_{AB} denote the wrapper oracle that takes (i, L_i) as input and sends (i, L_i) to \mathcal{O}_B if $i \in B$ -Labels, or to \mathcal{O}_{A1} if $i \in A$ -Labels and $i \in [\ell]$, or to \mathcal{O}_{A2} if $i \in A$ -Labels and $i \in [\ell+1, 2\ell]$. The oracle is accessible only once for each index *i*. Also, let \mathcal{O}_R be an oracle that evaluates a random function on given inputs. We consider Hash being correlation robust (as defined in Definition [17\)](#page-27-2) with respect to relations R_A and R_B . Hash is correlation robust if \mathcal{O}_{AB} and \mathcal{O}_R are indistinguishable.

We next define distribution of label sets by an algorithm, SmplLeaves, that samples an instance as follows. We also show an algorithm, SmplHalfLeaves, that simulates GetLeaf perfectly for specified half of the positions.

SmplLeaves(*P*) $\overline{01: \ell \leftarrow InputLen(P)}$ 02: ∀*i* ∈ [*ℓ*] 03: $L_i \leftarrow \{0,1\}^{\lambda}$ 04: if $i \in B$ -Labels 05: $L_{i+\ell} := L_i \oplus \Delta$ 06: if *i* ∈ A-Labels 07: $p \leftarrow \{0, 1\}^{\lambda}$ 08: $\pi := \text{LSB}(L_i) \oplus \text{LSB}(p)$ 09: $k := \text{KDF}(i, L_i) + (1 - 2\pi)\delta$ 10: $L_{i+\ell} := k||p|$ 11: Output $L := (L_1, \ldots, L_{2\ell})$ SmplHalfLeaves(*P, I*) $\overline{01: \ell \leftarrow InputLen(P)}$ 02: $I_1 := \{i \in I \mid i \leq \ell\}, I_2 := I \setminus I_1$ 03: ∀*i* ∈ *I*¹ 04: $L_i \leftarrow \{0,1\}^{\lambda}$ 05: ∀*i* ∈ *I*² 06: if *i* − *ℓ* ∈ B-Labels 07: $L_i \leftarrow \{0,1\}^{\lambda}$ 08: if *i* − *ℓ* ∈ A-Labels 09: $p \leftarrow \{0,1\}^{\ell}$ 10: $L_{i-\ell} \leftarrow \{0,1\}^{\lambda}$ 11: $\pi := \text{LSB}(L_{i-\ell}) \oplus \text{LSB}(p)$ 12: $k := \text{KDF}(i - \ell, L_{i-\ell}) + (1 - 2\pi)\delta$
13: $L_i := k||p$ $L_i := k||p$ 14: Output $L := (L_i)_{i \in I}$.

We use SmplLeaves to simulate O*[∆]* in the left-term game of the advantage function in Definition [17,](#page-27-2) and SmplHalfLeaves for \mathcal{O}_R in the right term. Note that in the right-term game, only the *I*-indexed part of *L* sampled from \mathcal{O}_R is used. Hence, SmplHalfLeaves suffices. We consider a Merkle tree algorithm that first hashes each of the given leaves by Hash and then computes binary hash trees towards the root. For leaf values *L*, let *HL* denote the hashed leaves. Let MT*.*Com′ and MT*.*Proof′ be respective Merkle tree algorithms that take hashed leaves *HL* as input instead of *L*. MT Correlation robustness games are unchanged but now described equivalently by separately hashing the inputs and applying MT*.*Com′ and MT*.*Proof′ .

Theorem 11. *Merkle tree scheme* MT *with* Hash *as above is MT correlation robust (as defined in Definition [15\)](#page-27-1)* with respect to correlations R_A and R_B if Hash *is correlation robust (as defined in Definition [17\)](#page-27-2) with respect to R^A and RB.*

Proof. We show that if there is an adversary, A, that distinguishes the MT correlation robustness games with the Merkle tree, we can construct algorithm β that distinguishes oracle \mathcal{O}_{AB} from \mathcal{O}_R , breaking the correlation robustness of Hash. For P and I , algorithm B works as follows.

 $Reduction$ $\mathcal{B}^{\mathcal{O}}$: 01: ℓ ← InputLen(P) 02: $I_1 := \{i \in I \mid i \leq \ell\}, I_2 := I \setminus I_1$ 03: $(L_i)_{i \in I}$ ← SmplHalfLeaves(P, I) 04: $\forall i \in I_1, h_{i+1} \leftarrow \mathcal{O}(i, L_i)$ 05: $\forall i \in I_2, h_{i-\ell} \leftarrow \mathcal{O}(i, L_i)$ 06: ∀*i* ∈ *I*, *hⁱ* = Hash(*Li*) 07: $HL := (h_i)_{i \in [2\ell]}$ 08: $h \leftarrow \textsf{MT}.\textsf{Com}'(HL)$ 09: $\pi \leftarrow \mathsf{MT}. \mathsf{Proof}'(HL,I)$ 10: Output $\mathcal{A}(h, \pi, L_{i \in I})$.

By construction of $\mathcal{B}^{\mathcal{O}}$, if $\mathcal{O} = \mathcal{O}_R$, \mathcal{B} produces HL of the right term of the MT correlation robustness game. If, on the other hand, $\mathcal{O} = \mathcal{O}_{AB}$, *HL* follows the distribution of hashing *L* given from SmplLeaves. Thus, it constitutes the view in the left term of the MT correlation robustness game. Accordingly, the advantage of B is the same as that of A. □

C ZKGC Framework [\[40\]](#page-24-0)

The protocol π_{ZK} in [\[40\]](#page-24-0) is shown in Figure [7.](#page-36-3) Let $\mathcal{G} = (\mathsf{Gb}, \mathsf{Ev}, \mathsf{Ve})$ be a garbling scheme. Let *L* be an NP language with $|w| < n = poly(|y|)$ and $f_y: \{0, 1\}^n \to$ $\{0,1\}$ be the verification function that outputs 1 if *w* is a valid witness for *y*. Both parties have input y and a security parameter 1^k . In addition, the prover *P* has input $w = (w_1, \dots, w_n)$.

- 1. For all $i \in [n]$, *P* sends (choose, *i*, *w_i*) to \mathcal{F}_{COT} ;
- 2. \mathcal{F}_{COT} sends *V* messages (chosen, *i*) (for all $i \in [n]$);
- 3. *V* runs $(GC, \{K_i^0, K_i^1\}_{i \in [n]}, Z) \leftarrow \mathsf{Gb}(1^k, f_y);$
- 4. For all $i \in [n]$, *V* sends to \mathcal{F}_{COT} the input (transfer, *i*, K_i^0, K_i^1);
- 5. \mathcal{F}_{COT} sends *P* messages (transferred, *i*, $K_i^{'}$) (for all $i \in [n]$);
- 6. *V* sends *GC* to *P*;
- 7. *P* runs $Z' \leftarrow \mathsf{Ev}(GC, \{K'_i\}_{i \in [n]})$; In case the function Ev aborts, set Z' to ⊥;
- 8. *P* sends (commit, 1, Z') to \mathcal{F}_{COT} , and \mathcal{F}_{COM} outputs (committed, 1, Z') to V ;
- 9. *V* sends the message (open-all) to the \mathcal{F}_{COT} functionality;
- 10. \mathcal{F}_{COT} sends P , for all $i \in [n]$, the values (transfer, *i*, K_i^0, K_i^1);
- 11. *P* runs $\mathsf{Ve}(GC, \{K_i^o, K_i^1\}_{i \in [n]})$, if the output is not accept, *P* terminates the protocol. Otherwise, if Ve outputs accept, P sends (reveal, 1) to $\mathcal{F}_\mathrm{COM};$
- 12. When *V* receives (reveal, 1, *Z'*) from $\mathcal{F}_{\text{COM}}, V$ outputs accept if $Z' \stackrel{?}{=} Z;$

Fig. 7: ZKGC protocol in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{COM}})$ -hybrid model in [\[40\]](#page-24-0). See [40] for details of the functionalities.

D Proofs

D.1 Proof of Theorem [1](#page-13-1)

Proof. (Correctness.) First observe that, since $LSB(x_i^0) \neq \text{LSB}(x_i^1)$, it holds that $LSB(k_i^{x_i}) = \pi_i \oplus x_i = \pi_i + x_i - 2\pi_i x_i$ for $\pi_i = LSB(k_i^0)$. Then, for every $i \in \mathsf{B}\text{-}\mathsf{Lables}(F)$, we have:

$$
\varphi_i = \text{KDF}(i, k_i^{x_i}) + \text{LSB}(k_i^{x_i}) \cdot t_i \n= (1 - x_i) \cdot K_i + x_i \cdot D_i + (\pi_i + x_i - 2\pi_i x_i) \cdot (1 - 2\pi_i) \cdot (K_i - D_i + \delta) \n= (1 - \pi_i) \cdot K_i + \pi_i \cdot D_i + (x_i - \pi_i)\delta \n= w_i^0 + x_i \delta.
$$

For $i \in$ A-Labels, $\varphi_i = w_i + x_i \delta$ holds. By letting $w^0 = (w_i^0)_i$ and $\varphi = (\varphi_i)_i$, we can write $\varphi = w^0 + \langle x \rangle \delta$. Then, for every *x* satisfying $\hat{g}_1 \odot x + \hat{g}_2 = y$, we have

$$
\hat{z} = \hat{g}_1 \odot (\mathsf{G} \cdot \varphi)
$$

= $\{\hat{g}_1 \odot (\mathsf{G} \cdot \langle x \rangle \cdot \delta)\} + \{\hat{g}_1 \odot (\mathsf{G} \cdot w^0)\}$
= $\{\hat{g}_1 \odot (\mathsf{G} \cdot \langle x \rangle \cdot \delta)\} + \hat{g}_2 \odot \delta + \{\hat{g}_1 \odot (\mathsf{G} \cdot w^0)\} - \hat{g}_2 \odot \delta$
= $y \odot \delta - \hat{g}_2 \odot \delta + \{\hat{g}_1 \odot (\mathsf{G} \cdot w^0)\}$
= d ,

and $\text{De}(\text{Ev}(\hat{F}, \text{En}(e, x)), d) = 1$. For *x* that $\hat{g}_1 \odot x + \hat{g}_2 \neq y$, $\hat{z} \neq d$ holds in the same transitions as above since $(\hat{g}_1 \odot x + \hat{g}_2) \odot \delta \neq y \odot \delta$. We thus conclude that $De(Ev(\hat{F}, En(e, x)), d) = 0.$

(Verifiable Correctness.) We construct VeC as follows.

- 1. Given (F, \hat{F}, e, d) , it first parses *e* into $(k_i^0, k_i^1)_i$. It then checks that there exists Δ that $\Delta = k_i^1 \oplus k_i^0$ for all $i \in \mathsf{B}\text{-}\mathsf{L}$ abels, and δ that $\delta = k_i^1 - k_i^0$ for all *i* ∈ A-Labels. It then checks that $\Delta \in \{0,1\}^{\lambda-1}$ ||1 and $\delta \in \mathcal{R}^*$.
- 2. Furthermore, it checks if $(\hat{F}, d) = \mathsf{Gb}_2(\hat{F}, e)$.
- 3. Output 1 if all the above checks pass. Output 0, otherwise.

If the checks in the first step pass, then $e \in \mathsf{Gb}_1(1^\lambda, F)$ and $(\hat{F}, d) = \mathsf{Gb}_2(F, e)$. From the perfect correctness, $\text{De}(\text{Ev}(\hat{F}, \text{En}(e, x)), d) = F(x)$ holds for all *x*. (Verifiability.) We construct VeE in the same way as VeC except that it drops

the check about *d* in the second step. We then construct ExtE as follows.

- 1. Given $F, \hat{F}, e, 1$ as input, parse e into $(k_i^0, k_i^1)_i$.
- 2. For every $i \in B$ -Labels, compute $\pi_i := \text{LSB}(k_i^0), K_i := \text{KDF}(i, k_i^0), D_i :=$ $KDF(i, k_i^1)$, and $w_i^0 := (1 - \pi_i) \cdot K_{i} + \pi_i \cdot (D_i - \delta)$.
- 3. For every $i \in A$ -Labels, set $w_i^0 := k_i^0$.
- 4. Set $\delta := k_i^1 k_i^0$ with arbitrary $i \in$ **A-Labels.**
- 5. Output $d = (\hat{g}_1 \odot (G \cdot w^0)) + ((y \hat{g}_2) \odot \delta).$

For (F, \hat{F}, e) that VeE outputs 1, there exists $\delta \in \mathcal{R}^*$ that $\delta = k_i^1 - k_i^0$ for all *i* ∈ A-Labels. Observe that ExtE computes w_i^0 and *d* in the same way as Gb₂ does. Thus, due to the correctness, $\mathsf{Ev}(\hat{F}, k^x) = \mathsf{Ext}(\hat{F}, e, 1)$ holds for all *x* satisfying $F(x) = 1.$

(Privacy.) We construct a separable privacy simulator (S_1, S_2) as illustrated below.

$$
\begin{array}{ll}\n\mathcal{S}_1(1^\lambda, F): \\
\hline\n\text{For } i \in \mathsf{B}\text{-Labels} \\
k_i \leftarrow \{0, 1\}^\lambda \\
\text{For } i \in \mathsf{A}\text{-Labels} \\
\text{For } i \in \mathsf{A}\text{-Labels} \\
k_i \leftarrow \mathcal{R} \\
\text{Output } \hat{x} := (k_i)_i.\n\end{array}\n\qquad\n\begin{array}{ll}\n\mathcal{S}_2(F, 1, \hat{x}): \\
(k_i)_i \leftarrow \hat{x}, \\
\text{For } i \in \mathsf{B}\text{-Labels} \\
t_i \leftarrow \mathcal{R}^* \\
\varphi_i := \text{KDF}(i, k_i) + \text{LSB}(k_i) \cdot t_i \\
\text{For } i \in \mathsf{A}\text{-Labels} \\
\varphi_i := k_i \\
d := \hat{g}_1 \odot (\mathsf{G} \cdot \varphi) \\
\text{Output } \hat{T} = (t_i)_{i \in [\ell]} \text{ and } d.\n\end{array}
$$

We begin with the privacy game with the simulator S_1 and S_2 as shown in Definition [8.](#page-11-0) It is the starting point, H_0 , of the hybrid we make. Let P_i denote the probability that the adversary outputs 1 in game *Hⁱ* . Note that the adversary is restricted to output *x* satisfying $F(x) = 1$.

In H_1 , we replace $\hat{x} \leftarrow S_1(1^{\lambda}, F)$ with $e \leftarrow \mathsf{Gb}_1(1^{\lambda}, F)$ and $\hat{x} \leftarrow \mathsf{Ev}(e, x)$. Since \hat{x} distributes uniformly over the appropriate domain in both H_0 and H_1 , we have $P_0 = P_1$.

In H_2 , we generate t_i as follows:

if
$$
x_i = 0
$$

\n $\pi_i := \text{LSB}(k_i), K_i := \text{KDF}(i, k_i), D_i \leftarrow \mathcal{R}^*$
\nelse
\n $\pi_i := 1 - \text{LSB}(k_i), K_i \leftarrow \mathcal{R}^*, D_i := \text{KDF}(i, k_i)$
\n $t_i := (1 - 2\pi_i)(K_i - D_i + \delta)$

Since either K_i or D_i is uniformly random, t_i remains uniform over \mathcal{R} . Thus, $P_1 = P_2.$

In H_3 , we replace uniform selections from R with oracle calls to O_4 . The difference, $P_3 - P_2$, is bound by the correlation robustness of KDF. It is noted that *∆* defined by O*[∆]* differs from the one defined by *e*. It, however, does not interfere with the reduction since $k_i^{\bar{x}_i}$ defined by *e* is independent of the view of the adversary, and so is *∆*.

In H_4 , parse \hat{x} as $\hat{x} = (k_i^{x_i})_i$, and set π_i , K_i , and D_i as

$$
\pi_i := \mathsf{LSB}(k^0_i), \, K_i := \mathsf{KDF}(i, k^0_i), \, D_i := \mathsf{KDF}(i, k^1_i).
$$

This change replaces *∆* defined by oracle O*[∆]* with the one defined by *e*. Since both distribute uniformly, the view of the adversary is unchanged. Thus $P_4 = P_3$.

In H_5 , we change φ_i and the way *d* is computed as follows:

For
$$
i \in \mathsf{B}\text{-Labels}
$$

\n $w_i^0 := (1 - \pi_i) \cdot K_i + \pi_i \cdot (D_i - \delta)$
\nFor $i \in \mathsf{A}\text{-Labels}$
\n $w_i^0 := k_i^0$
\n $d := (\hat{g}_1 \odot (\mathsf{G} \cdot w^0)) + ((y - \hat{g}_2) \odot \delta)$

As shown in the proof of correctness, this change yields the same *d* when *x* satisfies $F(x) = 1$ (it is indeed the case we consider). Hence, $P_5 = P_4$.

In H_6 , we replace $(\tilde{F}, d) \leftarrow \mathcal{S}_2(F, 1, \hat{x})$ with $(\tilde{F}, d) \leftarrow \mathsf{Gb}_2(F, e)$. Since modified \mathcal{S}_2 is the same as Gb_2 , their output (\hat{F}, d) distributes identically. Hence, $P_6 = P_5$.

Now, H_6 is the same as the privacy game with the real garbling scheme. By accumulating the above bounds, we conclude that the garbling scheme is private with respect to *x* satisfying $F(x) = 1$ if KDF is correlation robust.

(Soundness and Output Indistinguishability.) Recall the soundness game defined in Definition [5](#page-11-1) where $(\hat{F}, e, d) \leftarrow$ Gb $(1^{\lambda}, F)$ is run and adversary $\mathcal A$ is given \hat{F} and $\hat{x} := \text{En}(e, x)$ for *x* satisfying $F(x) = 0$. Call it game H_0 and let P_0 denote the probability that A outputs *d*. Let *e* parse into $(k_i^0, k_i^1)_i$.

In H_1 , we modify Gb_2 by replacing $\mathsf{KDF}(i, k_i^{1-x_i})$ for each $i \in \mathsf{B}\text{-}\mathsf{labels}$ with randomly and independently chosen values in R . A does not notice the change due to the correlation robustness of KDF. Namely, $P_1 - P_0$ is bound by the advantage in the correlation robustness of KDF.

Observe that, in H_1 , t_i for every $i \in \mathsf{B}\text{-}\mathsf{L}$ abels distributes uniformly due to the randomness used in place of $\text{KDF}(i, k_i^{1-x_i})$. Also, \hat{x} distributes uniformly over its appropriate domain. Hence, the view of A is independent of δ .

For w_i^0 in Gb_2 , let $w_i^1 = w_i^0 + \delta$. Observe that, for x^* satisfying $f(x^*) = y$,

$$
d = (\hat{g}_1 \odot (\mathsf{G} \cdot w^0)) + ((y - \hat{g}_2) \odot \delta)
$$

= (\hat{g}_1 \odot (\mathsf{G} \cdot w^0)) + (\hat{g}_1 \odot (\mathsf{G} \cdot \langle x^* \rangle \cdot \delta))
= \hat{g}_1 \odot (\mathsf{G} \cdot (w^0 + \langle x^* \rangle \cdot \delta))
= \hat{g}_1 \odot (\mathsf{G} \cdot w^{x^*})

where $w^x = (w_i^{x_i^*})_i$.

Given $\hat{x} = w^0 + \langle x \rangle \delta$ for any *x* of $f(x) \neq y$, the adversary sees $\hat{g}_1 \odot (\mathsf{G} \cdot \hat{x}) =$ $\hat{g}_1 \odot (\mathsf{G} \cdot w^0) + \hat{g}_1 \odot (\mathsf{G} \cdot \langle x \rangle \cdot \delta) = \hat{g}_1 \odot (\mathsf{G} \cdot w^0) + (y - \hat{g}_2) \odot \delta + \hat{g}_1 \odot (\mathsf{G} \cdot \langle x \rangle \cdot \delta) - (y - \hat{g}_2) \odot \delta = 0$ $(d) + (f(x) - y) \odot \delta$. Observe that, for \hat{x} and any δ , there exists w^0 that satisfies $\hat{x} := w^0 + \langle x \rangle \cdot \delta$. Therefore, for $d = \hat{g}_1 \odot (\mathsf{G} \cdot \hat{x}) - (f(x) - y) \odot \delta$, we can think that the first term, $\hat{g}_1 \odot (\mathsf{G}\hat{x})$, is fixed, independent of δ . Also observe that operation \odot is injective for $f(x) - y \neq 0$. Thus, *d* follows the distribution of δ , and event $d' = d$ happens only with probability $1/|\mathcal{R}^*|$ over the choice of δ . Accordingly, it is sound with soundness error $1/|\mathcal{R}^*|$. As it is shown that *d* is independent of the view of the adversary, it is indistinguishable from one uniformly sampled from $\mathcal{D} = \{d \mid d := (\hat{g}_1 \odot (\mathsf{G} \cdot w^0)) + ((y - \hat{g}_2) \odot \delta), \delta \in \mathcal{R}^*\}.$ Thus, the scheme is output indistinguishable for *x* satisfying $F(x) = 0$. □

D.2 Proof of Theorem [3](#page-16-1)

Proof. As Gb_1 and En are assumed to be common for B and A, the correctness and soundness of U are directly reduced from the respective properties of B and A. For verifiability, U*.*VeE and U*.*ExtE are obtained simply by executing corresponding algorithms of B and A in parallel as below. Proof is done by straightforward reduction from the verifiability of B and A.

⊓⊔

D.3 Proof of Theorem [4](#page-16-2)

Proof. (Correctness) Let *Γ* be the access structure implied by P_0 . For every $x \in$ $\{0,1\}^{\ell}$, let $T := \{i | P_i(x)\}$. Due to correctness of P_i , P_i . Ev(\hat{P}_i , En(e, x)) = d_i for

all $i \in T$. Let $t'_i := \mathsf{SE}.\mathsf{Dec}_{\mathsf{KDF}_i(d_i)}(c_i)$. Due to correctness of $\mathsf{SE}, t'_i = t_i$ for $i \in T$. $\text{For } T \in \Gamma, \text{SS}.\text{Rec}(P_0, t'_{i \in [T]}) = d \text{ holds and, for } T \notin \Gamma, \text{SS}.\text{Rec}(P_0, t'_{i \in [T]}) \neq d$ holds due to the correctness of SS.

(Soundness) Starting from the soundness game as in Definition [5,](#page-11-1) we construct a hybrid of games changing how \hat{P} is prepared.

 H_0 : The soundness game where, for legitimately generated (\hat{P}, e, d) and some fixed *x* that $P(x) = 0$, *A* is given \hat{P} and $\hat{x}' := M.En(e, x)$ as input. It then outputs \tilde{d} . Let pr_0 be the probability that $\tilde{d} = d$ happens. In the following, pr_i denotes the probability that $\tilde{d} = d$ happens in H_i . Let $T := \{i | P_i(x)\}.$

H₁: For each $i \notin T$, replace d_i with a random value sampled from decoding key domain \mathcal{D}_i for *x*. This change is bound by the output indistinguishability of P_i . Thus, the difference from pr_0 to pr_1 is negligible.

 H_2 : For each $i \notin T$, replace $KDF_i(d_i)$ with a random value over key space $\{0,1\}^{\lambda}$ of SE. This change is bound by the indistinguishability of KDF*ⁱ* . Thus, the difference from pr_1 to pr_2 is negligible.

H₃: For each $i \notin T$, replace t_i with a random value of the same length. This change is bound by the CPA security of **SE**. Thus, the difference from pr_2 to pr_3 is negligible.

We then claim that in H_3 , the probability of the adversary correctly guessing *d* is bound by the security of SS. Accumulating all the above bounds, we conclude that A in the original soundness game outputs *d* only with negligible probability.

(Verifiability) We construct M*.*VeE and M*.*ExtE as follows.

 $M.VeE(\hat{P}, P, e)$: $\overline{(\hat{P}_i, c_i)_{i \in [n]} + \hat{P}},$ $(P_0, P_{i \in [n]}) \leftarrow P$, $∀ i ∈ [n]$ $b_i \leftarrow \mathsf{P}_i.\mathsf{VecE}(P_i, \hat{P}_i, e)$ $d_i \leftarrow \mathsf{P}_i$.Ext $\mathsf{E}(\hat{P}_i, e, 1)$ $t_i \leftarrow \mathsf{SE}.\mathsf{Dec}_{\mathsf{KDF}_i(d_i)}(c_i)$ $b_0 \leftarrow$ SS.Ve $(P_0, t_{i \in [n]})$ Output $\wedge_{i=0}^n b_i$. $M.ExtE(\hat{P}, e, 1)$: $\overline{(\hat{P}_i, c_i)_{i \in [n]} + \hat{P}},$ ∀ *i* ∈ [*n*], $d_i \leftarrow \mathsf{P}_i$.Ext $\mathsf{E}(\hat{P}_i, e, 1)$ $t_i \leftarrow \mathsf{SE}.\mathsf{Dec}_{\mathsf{KDF}_i(d_i)}(c_i).$ $d \leftarrow$ SS.Rec($P_0, t_{i \in [n]}$). Output *d*.

We first argue that M.VeE outputs 1 for honestly generated (\hat{P}, e) . Every $P_i.\mathsf{Vec}(P_i, \hat{P}_i, e)$ returns $b_i = 1$ for $e \leftarrow \mathsf{Gb}_1(1^{\lambda}, P_1)$ and $(\hat{P}_i, d_i) \leftarrow \mathsf{P}_i \cdot \mathsf{Gb}_2(P_i, e)$, and P_i . Ext $\mathsf{E}(\hat{P}_i, e, 1)$ outputs $d'_i = d_i$ if P_i is satisfiable. Then, d_i decrypts correctly formed ciphertext c_i and recovers share t_i . As the shares are correctly made through SS.Share, SS.Ve($P_0, t_{i \in [n]}$) returns $b_0 = 1$. As each step works except for negligible probability, M*.*VeE outputs 1 except for negligible probability as well.

We then show that if M*.*VeE outputs 1, then M*.*ExtE outputs *d* that decodes to 1. Observe that if M.VeE accepts (\hat{P}, P, e) , then, due to the verifiability of SS, there exists unique *d* that $(\hat{P}, e, d) \in M$. Gb $(1^{\lambda}, P)$. For such \hat{P} and *e*, every P_i **ExtE** in M**.ExtE** recovers correct d_i , which in turn recovers share t_i that passes SS*.*Ve. Then SS*.*Rec on the shares recovers *d*, which is the encoding of output 1. ⊓⊔

D.4 Proof of Theorem [5](#page-17-2)

Proof. Since correctness can be verified by inspection, we focus on zero-knowledge and soundness. We first construct a special honest verifier zero-knowledge simulator, S_{ZK} , using privacy simulator S_{G} and hiding simulator S_{S} as defined in Definition [8](#page-11-0) and Definition [20,](#page-30-1) respectively as follows.

 $\mathcal{S}_{\mathsf{ZK}}(CH)$: $01\colon s\leftarrow \{0,1\}^{\lambda}$ 02: $RT := (s, M)$ 03: $(s_i)_{i \in [M]}$ ← N.Expand(RT) 04: *ST* ← N.SubTrees $(RT, [M] \setminus CH)$ 05: ∀*i* ∈ [*M*] \ *CH* 06: $(e_i, h_i) \leftarrow \mathsf{S}. \mathsf{ComGb}_1(1^\lambda; s_i)$ $07:$ $(\hat{P}_i, d_i) \leftarrow G.Gb_2(P, e_i)$ 08: ∀*i* ∈ *CH* 09: $(\hat{P}, \tilde{x}, d) \leftarrow \mathcal{S}_{\mathsf{G}}(1^{\lambda}, P, 1)$ 10: $(h_i, \sigma_i) \leftarrow \mathcal{S}_{\mathsf{S}}(\tilde{x})$ 11: *H* ← Hash $((\hat{P}_i, d_i, h_i)_{i \in [M]})$ 12: Output *H*, *ST*, $(\hat{P}_i, d_i, \tilde{x}_i, \sigma_i)_{i \in CH}$.

We then argue that the output from S_{ZK} is indistinguishable from the real one. As usual, we follow the hybrid argument. Let H_0 be the process where S_{ZK} is executed and transcript $(H, ST, (\hat{P}_i, d_i, \tilde{x}_i, \sigma_i)_{i \in CH})$ is produced. We consider how the output distribution changes as we modify S_{ZK} .

In H_1 , we replace simulator S_G in line 09 with the real process

09-1:
$$
e_i \leftarrow \text{G.Gb}_1(1^{\lambda}, P; s'_i)
$$

09-2: $\tilde{x} \leftarrow \text{S.CompEn}(P, s'_i, x)$
09-3: $(\hat{P}_i, d_i) \leftarrow \text{G.Gb}_2(P, e_i)$

using the real witness x and uniformly random s'_{i} . This change is negligible, i.e., the output distribution in H_0 and H_1 are indistinguishable due to the privacy of G for *x* satisfying $P(x) = 1$.

In H_2 , we remove simulator S_5 in line 10, and replace $G.Gb_1$ and $S.GompEn$ in 09-1 and 09-2 as follows:

$$
\begin{array}{l} \texttt{09-1: } (e_i, h_i) \leftarrow \texttt{S}. \texttt{ComGb}_1(1^\lambda, P; s'_i) \\ \texttt{09-2: } (\tilde{x}, \sigma_i) \leftarrow \texttt{S}. \texttt{ProvEn}(P, s'_i, x) \end{array}
$$

This change is negligible due to the hiding property of S.

In H_3 , we replace all s'_i in line 09-1 with s_i generated by N.Expand in line 03. This change is negligible due to the pseudo-randomness of N.

By rearranging the lines appropriately, we observe that the modified simulator in *H*³ does precisely the same as the real prover algorithm. It completes the proof of special honest verifier zero-knowledge.

Next, we show that the protocol is special sound for the extended statement. Suppose that two accepting transcripts having the same initial message *H* with different challenges are given. Observe that there exists at least one session, *i*^{*}, that *i*^{*} is in the first challenge but not in the second one. Namely, *s*_{*i*}∗ and $(\hat{P}_{i^*}, d_{i^*}, \tilde{x}_{i^*}, \sigma_{i^*})$ are obtained from the transcripts.

We construct extractor *E* that, given the transcripts, either outputs witness *x* satisfying $P(x) = 1$, or a collision for Hash, or (s, \tilde{x}, σ) breaking binding of S, or an evidence that G is not verifiably correct. *E* works as follows.

- [1.](#page-14-0) It verifies the transcripts as the verifier does in Figure 1. If $(\hat{P}_i, d_i, h_i)_{i \in [M]}$ obtained from the transcripts differ, output them and halt. The output forms a collision of Hash since both satisfies $H =$ Hash $((\hat{P}_i, d_i, h_i)_{i \in [M]})$ for the same first message H.
- 2. Compute $x_{i^*} \leftarrow S$.Comp $En^{-1}(P, s_{i^*}, \tilde{x}_{i^*})$. If $x_{i^*} = \perp$, output $(s_{i^*}, \tilde{x}_{i^*}, \sigma_{i^*})$ and halt. The output breaks the binding of S since $h_{i^*} \leftarrow S.VerCom(\tilde{x}_{i^*}, \sigma_{i^*})$ holds for $(e_{i^*}, h_{i^*}) \leftarrow \mathsf{S}.\mathsf{ComGb}_1(1^\lambda, P; s_{i^*}).$
- 3. If $P(x_{i^*}) = 0$, output $(P, \hat{P}_i, e_{i^*}, d_{i^*})$ and halt. Observe that $1 = G.VeC(P, \hat{P}_{i^*},$ (e_{i^*}, d_{i^*}) and $1 = \mathsf{G}.\mathsf{De}(\mathsf{G}.\mathsf{Ev}(\hat{P}_{i^*}, \hat{x}_{i^*}), d_{i^*})$ holds. Also observe that, in the previous step, S.CompEn⁻¹ computes x_{i^*} by x_{i^*} ← G.En⁻¹(e, \hat{x}_{i^*}). Accordingly, $P(x_{i^*}) = 0$ implies $G.De(G.Ev(\hat{P}_{i^*}, G.En(e, x_{i^*}), d_{i^*}) \neq P(x_{i^*}),$ breaking the verifiable correctness of G.
- 4. Output x_i ^{*}. This satisfies $P(x_i^*) = 1$.

It is obvious that above extractor *E* works in polynomial time. This completes the proof of special soundness for the extended statement. ⊓⊔

D.5 Proof of Theorem [6](#page-20-1)

Proof. The correctness hold trivially from the correctness of B and A as Gb_1 and En are common in B and A.

For verifiable correctness and privacy, we construct the verification algorithm, VeC, and the privacy simulator S as follows:

From the construction of U*.*VeC, verifiable correctness is directly obtained by that of B and A.

For proving privacy, we follow the hybrid argument starting from the privacy game with the above simulator. Let H_0 denote the game and P_0 be the probability that the adversary outputs 1.

In H_1 , we replace $B.S_1$ and $B.S_2$ with $e \leftarrow B.Gb_1(1^{\lambda}, C)$, $\hat{x} \leftarrow B.Ev(e, x)$, and $(\hat{C}, d_{\mathsf{B}}) \leftarrow \mathsf{B}.\mathsf{Gb}_2(C, e)$. The difference, $P_0 - P_1$, is bound by the privacy of B.

In *H*₂, we replace A.S₂ with (\hat{F}, d_A) ← A.Gb₂(*F, e*). The difference, $P_2 - P_1$, is bound by the privacy of A.

Game H_2 is the same as the privacy game with the real garbling scheme. Accordingly, by observing that the above differences are all negligible if B and A are private with respect to *x* satisfying $C(x) = 1$ and $F(x) = 1$, we conclude that U is private with respect to *x* respecting the same restriction. $□$

D.6 Proof of Theorem [7](#page-20-2)

Proof. (Correctness) From the correctness of A and B, and their out-in-compatibility, we have:

$$
U.De(U.Ev(\hat{P}, U.En(e, x)), d)
$$

= A.De(A.Ev(\hat{F}, B.Ev(\hat{C}, B.En(e, x))), d)
= A.De(A.Ev(\hat{F}, A.En(d_B, C(x))), d)
= F \circ C(x)

(Verifiable Correctness) We construct U*.*VeC as follows.

$$
\begin{array}{l} \underbrace{\mathsf{U}.\mathsf{Vec}(1^\lambda,P,\hat{P},e,d)\mathsf{:}}_{\text{$(C,F)\text{ \leftharpoonup}\mathsf{P},~\text{$(\hat{C},\hat{F})\text{ \leftharpoonup}\mathsf{P}$}}\\ \underbrace{\mathsf{b}_\mathsf{B}\leftarrow\mathsf{B}.\mathsf{VeD}(C,\hat{C},e)}_{d_\mathsf{B}\leftarrow\mathsf{B}.\mathsf{ExtD}(\hat{C},e)}\\ b_\mathsf{A}\leftarrow\mathsf{A}.\mathsf{Vec}(F,\hat{F},d_\mathsf{B},d)\\ \text{Output $(b_\mathsf{B}\land b_\mathsf{A})$?} \end{array}
$$

We first show that $U.VeC$ returns 1 for honestly generated inputs. Let $(\hat{C}, e, d_\mathsf{B}, \hat{F}, d)$ be those observed in running U.Gb. Suppose that U.VeC is run on input $(C, F, \hat{C}, \hat{F}, \hat{C})$ *e, d*). Due to the decode key extractability of B, B*.*VeD returns 1 and B*.*ExtD outputs d_{B} except for a negligible probability. Then, due to the verifiable correctness of A, A.Ve $\mathsf{C}(F, \hat{F}, d_\mathsf{B}, d)$ outputs 1. Thus, U.VeC returns 1 for honestly generated inputs.

Next we argue that, whenever U.VeC returns 1, $\text{U.De}(\text{U.Ev}(\hat{P}, \text{U.En}(e, x)), d)$ = $P(x)$ holds for all *x*. Suppose that, for given *C*, \hat{C} , and *e*, U.VeC returns 1 and B. ExtD outputs d_{B} . Then, due to the decode key extractability of B, it holds that $C(x) = B \cdot De(B E v(\hat{C}, B \cdot En(e, x)), d_B)$ for any *x*. Consider $U.Ev(\hat{P}, U \cdot En(e, x))$. From the construction of U.Ev and U.En, it is equivalent to $\mathsf{A}.\mathsf{Ev}(\hat{F}, \mathsf{B}.\mathsf{Ev}(\hat{C}, \mathsf{B}.\mathsf{En}(e,$ *x*))). As we assume the decoding of B and encoding of A are compatible, it holds that $A.En(d_B, B.De(\hat{y}, d_B)) = \hat{y}$ for any \hat{y} . Thus, we have

$$
U.Ev(\hat{P}, U.En(e, x))
$$

= A.Ev(\hat{F}, B.Ev(\hat{C}, B.En(e, x)))
= A.Ev(\hat{F}, A.En(d_B, B.De(B.Ev(\hat{C}, B.En(e, x)), d_B)))
= A.Ev(\hat{F}, A.En(d_B, C(x))

Finally, from the verifiable correctness of A, we have

$$
U.De(U.Ev(\hat{P}, U.En(e, x)), d)
$$

= A.De(A.Ev(\hat{F}, A.En(d_B, C(x)), d)
= F \circ C(x)

as expected. Thus, U is verifiably correct.

(Privacy) Using obliviousness simulator B*.*S and separable privacy simulator A. S_2 , we construct a privacy simulator $\cup \mathcal{S}$ as follows.

$$
\frac{\mathsf{U.S}(1^{\lambda}, P, 1):\n(C, F) \leftarrow P\n}{(\hat{C}, \hat{x}) \leftarrow \mathsf{B.S}(1^{\lambda}, C)\n \hat{y} \leftarrow \mathsf{B.Ev}(\hat{C}, \hat{x})\n \quad (\hat{F}, d) \leftarrow \mathsf{A.S}_2(F, 1, \hat{y})\n \quad \text{Output } \hat{P} := (\hat{C}, \hat{F}), \hat{x}, \text{ and } d.
$$

 H_0 : the privacy game with the above simulator.

*H*₁: Replace B*.S* with $(\hat{C}, e, d_{\mathsf{B}}) \leftarrow \text{B.Gb}(1^{\lambda}, C)$ and $\hat{x} \leftarrow \text{B.En}(e, x)$. $P_1 - P_0$ is bound by the obliviousness of B.

*H*₂: Replace $A.S_2(F, 1, \hat{y})$ with $A.Gb_2(F, d_B)$. $P_2 - P_1$ is bound by the privacy of A.

Aggregating the above transitions, we can conclude that U is private if B is oblivious and A is private. Since the privacy of A holds only for *y* satisfying $F(y) = 1$, U preserves privacy only for *x* satisfying $P(x) = F(C(x)) = 1$. □

D.7 Proof of Lemma [2](#page-21-1)

Proof. Recall that for any monotone function $f, x \leq y$ implies $f(x) \leq f(y)$. From the rightmost clause of *S*['], we have $R_i(x_i) = 1$ if $s_i = 1$, which means $s_i \leq R_i(x_i)$ holds for every *i*. Therefore, $C(s_1, \ldots, s_\ell) \leq C(R_1(x_1), \ldots, R_\ell(x_\ell))$ holds. Since *S'* is satisfied, $C(s_1, \ldots, s_\ell) = 1$. Thus, $C(R_1(x_1), \ldots, R_\ell(x_\ell)) = 1$, and *S* is satisfied as well. □

D.8 Proof of Theorem [8](#page-22-0)

Proof. Completeness is verified by inspection. It is knowledge sound as argued as follows. From Π_C , "switches" s_1, \ldots, s_n that open commitments c_1, \ldots, c_ℓ and satisfies $C(s_1, \ldots, s_n) = 1$ are extracted. From each $\Pi_{R_i \vee s_i}$, either valid witness x_i for R_i , or $s'_i = 0$ is extracted. For those indices *i* that $s'_i = 0$, $s_i = s'_i$ holds if com_{*i*} is binding. For every index *i* that $s_i = 1$, we've got x_i that $R_i(x_i) = 1$. For every index *i* that $s_i = 0$, set x_i an arbitrary value. Such x_1, \ldots, x_ℓ and s_1, \ldots, s_ℓ obviously fulfills *S*^{\prime}. For zero-knowledge, we build a simulator that sets $s_i = 0$ for all *i* and executes $\Pi_{R_i \vee s_i}$ honestly using the right-hand side of the disjunction. Then, given commitments c_i for all $i \in [n]$, the simulator runs zero-knowledge simulator of Π_C . One can then apply a hybrid argument to prove that the view created by this simulator is indistinguishable from the real one due to zero-knowledge of Π_C and witness indistinguishability of $\Pi_{R_i\vee s_i}$ and the hiding property of the commitment schemes.

³ For $x, y \in \{0, 1\}^{\ell}, \leq$ is a partial ordering. We write $x \leq y$ if and only if $x_i \leq y_i$ for every $i \in [\ell]$.