Efficient Adaptively Secure Zero-knowledge from Garbled Circuits

Chaya Ganesh[1], Yashvanth Kondi[2], Arpita Patra[3], and Pratik Sarkar[4]

[1] Department of Computer Science, Aarhus University. ganesh@cs.nyu.edu
[2] Northeastern University. ykondi@ccs.neu.edu
[3] Indian Institute of Science, India. arpita@iisc.ac.in
[4] Indian Institute of Science, India. pratiks@iisc.ac.in

Abstract. Zero-knowledge (ZK) protocols are undoubtedly among the central primitives in cryptography, lending their power to numerous applications such as secure computation, voting, auctions, and anonymous credentials to name a few. The study of efficient ZK protocols for non-algebraic statements has seen rapid progress in recent times, relying on the techniques from secure computation. The primary contribution of this work lies in constructing efficient UC-secure constant round ZK protocols from garbled circuits that are secure against *adaptive* corruptions, with communication linear in the size of the statement. We begin by showing that the practically efficient ZK protocol of Jawurek et al. (CCS 2013) is adaptively secure when the underlying oblivious transfer (OT) satisfies a mild adaptive security guarantee. We gain adaptive security with little to no overhead over the static case. A conditional verification technique is then used to obtain a three-round adaptively secure zero-knowledge argument in the non-programmable random oracle model (NPROM).

We draw motivation from state-of-the-art non-interactive secure computation protocols and leveraging specifics of ZK functionality show a two-round protocol that achieves static security. It is a proof, while most known efficient ZK protocols and our three round protocol are only arguments.

1 Introduction

Zero-knowledge (ZK) proofs introduced in [GMR85] provide a powerful tool in designing a variety of cryptographic protocols. Since then, they have been an important building block in various applications. Zero-knowledge proofs allow a prover to convince a verifier about the validity of a statement, while giving no information beyond the truth of the statement. Informally, an honest prover should always convince a verifier about a true statement (completeness). Moreover, a malicious verifier learns nothing beyond the validity of the statement (zero-knowledge) and a malicious prover cannot convince a verifier of a false statement (soundness). In addition to soundness, a ZK protocol in which the prover's witness can be extracted by a simulator offers *proof of knowledge*.

It is known that every language in NP has a zero-knowledge proof system [GMW86]. Despite this, proving generic statements is inefficient in practice, and there are few techniques that allow efficient proofs. These techniques almost always apply to a restricted set of languages, with a series of works [Sch90,GQ88,CM99,GS08] on proving algebraic relationships like knowledge of roots, discrete logarithms etc.

Kilian's zero-knowledge argument [Kil92] achieves sub-linear communication, but relies on PCP and is of theoretical interest. Groth [Gro10] gave the first constant-size non-interactive ZK proofs. Since then, many constructions of SNARKs (Succinct non-interactive arguments of knowledge) have been presented [GGPR13,Lip13,DFGK14,Gro16], and have been implemented as well [PHGR13,CFH+15]. Though SNARKs have short proofs and allow efficient verification, they have shortcomings in prover efficiency. The prover performs public-key operations proportional to the size of the circuit representing the statement. In addition, they rely on a large trusted parameter; for example, a long common reference string (CRS).

Around the same time that ZK was introduced, Yao introduced secure two-party computation (2PC) and garbled circuits (GC) [Yao82]. The problem of general multi-party computation (MPC) [Yao86,GMW87,BGW88] considers a set of parties holding private inputs with the task of computing a joint function while preserving certain desired security properties. An interesting line of recent works [IKOS07,BP12,JKO13,HMR15,CGM16,GMO16,HV16,AHIV17] establishes connections between MPC and ZK, and use the techniques of 2PC and MPC for truly efficient ZK protocols. The two main streams of works connecting MPC with efficient ZK protocols rely on "MPC-in-the-head" approach [IKOS07,IKOS09] and garbled circuit based approach [JKO13], as elaborated below.

1.1 Efficient ZK Protocols

Ishai et al. [IKOS07,IKOS09] show how to use an MPC protocol to obtain a ZK proof for an NP relation in the commitment-hybrid model. This approach, called "MPC-in-the-head", provides a powerful tool to obtain black-box constructions for generic statements without relying on expensive Karp reductions. Recently, this technique spurred progress in constructing practical ZK protocols [GMO16,CDG+17] resulting in efficient ZK arguments tailored for Boolean circuits, known as 'ZKBoo' and 'ZKBoo++' respectively. They study variants of the "MPC-in-the-head" framework, plug in different MPC protocols, and provide concrete estimates of soundness. In yet another recent attempt, [AHIV17] proposes 'Ligero', a 4 round interactive ZK argument with sub-linear (in the circuit size) proof-size relying on interactive PCPs and plugging in a refined MPC of [DI06] in the "MPC-in-the-head" approach. Specifically, they achieve a proof size of $\mathcal{O}(\lambda\sqrt{|C|\log|C|})$. The construction uses Reed Solomon Codes from coding theory techniques. The marked improvement in the proof size is obtained by careful tweaking of the protocol parameters. The prover and verifier time is $\mathcal{O}(|C|\log|C|)$ symmetric key operations, and without any public key operations. The protocol does not require any setup and the security is proven in the stand-alone setting. The constructions of [GMO16,CDG+17,AHIV17] can be made non-interactive using the Fiat-Shamir heuristic in the programmable RO model.

Jawurek et al. [JKO13] construct a UC-secure ZK protocol (referred to as ZKGC henceforth) using garbled circuits as the primary building block. The communication required for their protocol is linear in the size of the circuit implementing the NP relation, and is also concretely efficient as it achieves malicious security with only one garbled circuit. However, the protocol is inherently interactive. ZKGC is essentially a version of Yao's original constant-round 2PC protocol where the GC constructor has no input; this yields full malicious security at little overhead over the semi-honest case as Yao's protocol in this case is already secure against a malicious evaluator. The protocol uses oblivious transfer (OT). The use of OT in ZK protocols dates back to [KMO89]. Notably, Zero-knowledge, when viewed as a special case of 2PC, allows for a relaxation in the properties required of the underlying GCs, as noted in [JKO13]. This led to the introduction of the notion of *privacy-free* garbling schemes [FNO15], which are optimized for the ZK setting of [JKO13]. A privacy-free garbling scheme only achieves authenticity, and leverages privacy-freeness in order to save on communication and computation costs of garbling. Privacy-free GCs are further studied by Zahur et al. [ZRE15], who construct a privacy-free scheme using the HalfGates approach. Their privacy-free scheme makes use of FreeXOR [KS08] to garble and evaluate XOR gates at no cost, and produces only one ciphertext when garbling an AND gate (along with two calls to a hash function H). Their construction comprises the current state-of-the-art in privacy-free garbling for circuits. When formulaic circuits are of concern, [KP17] shows how to do privacy-free garbling with *zero* ciphertext and with information-theoretic security.

The interactive schemes based on garbled circuits allow for the flexibility of how the keys for the underlying GCs are constructed and how the garbled input (ie. witness) is encoded. This leads to interesting applications making non-blackbox use of ZKGC [CGM16,KKL$^+$16]. For instance, Kolesnikov et al. [KKL$^+$16] introduce a new primitive called "attribute selective encryption" as a method of input encoding in ZKGC in order to construct attribute-based key-exchange. This allows a client to prove to a server that it holds a certificate corresponding to its attributes issued by a trusted authority, and that these attributes satisfy a policy constructed by the server. Note that only proving knowledge of attributes satisfying a given policy is insufficient in this setting. Another point of comparison is that the PROM assumption required by non-interactive 'MPC-in-the-head' based ZK protocols can be used to construct highly efficient adaptively secure garbled circuits [BHR12a] allowing ZKGC and our protocol to be cast in the online-offline paradigm, with all circuit-dependent communication moved to a preprocessing stage.

Lastly, we note that all of the above protocols deal with *static* adversaries, where the adversary is allowed to choose the party it wishes to corrupt only at the outset of the protocol. In this work, we are interested in building efficient concurrently composable ZK protocols that can tolerate adaptive adversaries [Bea96a,CFGN96]. In the following section, we summarize the literature on practical ZK protocols for non-algebraic statements, and zero-knowledge protocols secure against adaptive adversaries.

1.2 Adaptively Secure Zero-knowledge

An adaptive adversary may dynamically decide which party to corrupt as the protocol progresses. Its choice of corruptions may be adapted according to the specific information it sees, possibly even corrupting both the parties. Tolerating an adaptive adversary in a ZK protocol in the UC setting requires a straight-line simulator that can generate a transcript on behalf of the prover without knowledge of the witness, and later be able to "explain" the transcript for any given witness (ie. concoct valid-looking corresponding local randomness). In [Bea96a], the authors show that the zero-knowledge proof system of GMW [GMW91] is not secure against adaptive adversaries or else the polynomial hierarchy collapses, and proceed to build ZK arguments. This work is further advanced in [CLOS02] where UC-secure ZK arguments are presented relying on adaptive commitments schemes. In [LZ11], it is shown that adaptive ZK proofs exist for all of NP assuming only one-way functions. They present constructions of adaptively secure ZK proofs from adaptive instance dependent commitment schemes.

Adaptive ZK via Adaptive MPC. The recent work of Cannetti et al. [CPV17] shows how to construct constant-round two party computation using garbled circuits in the standard model. They solve the problem of equivocating a garbled circuit in order to explain the view of a constructor who has already sent a GC in Yao's protocol by means of a *functionally equivocal* encryption scheme. However this comes at the cost of a GC whose size is quadratic in the size of the circuit that is garbled. Previous adaptively secure constant round secure computation protocols have relied on obfuscation [DSKR15,CGP15,CP16].

Adaptive ZK from MPC-in-the-head Approach. We note that the "MPC-in-the-head" approach is likely to generate adaptively secure ZK protocols by relying on adaptive commitments and possibly adaptively secure MPC. An adaptive commitment scheme is used to commit to the views of the virtual parties. The adaptive commitment schemes from standard assumptions [HV16,HPV17] may be taxing in terms of both communication and round efficiency. Alternatively, the commitments used in IKOS-style protocols can be implemented in the programmable random oracle model, allowing the simulator to equivocate committed views, which yields adaptive security in a straightforward manner. Another related method is via non-committing encryption (NCE), an approach that has in other circumstances allowed circumvention of known lower bounds in the plain model. For instance, the adaptively secure garbling scheme of [BHR12a] uses a programmable RO to achieve NCE, which results in the circumvention of a lower bound in the online communication complexity of adaptively secure garbling schemes shown by Applebaum et al. [AIKW15].

Adaptive ZK via 2PC-in-the-Head [HV16]. The work of [HV16] uses the "MPC-in-the-head" technique [IKOS09] to construct adaptive ZK proofs. Their use of interactive hashing [NOVY98] to construct instance dependent commitments to equivocate committed views requires a non-constant number of rounds. The overall round complexity of their adaptive ZK protocol is $\mathcal{O}(\mu \log \mu)$, where μ is the soundness parameter. The proof size is $\mathcal{O}(\mu|\mathbf{C}|\mathsf{poly}(\lambda))$ and the $\mathsf{poly}(\lambda)$ factor is $\Omega(\lambda)$. While their scheme can be made constant round by plugging in the appropriate instance-dependent commitment scheme, it comes at the cost of proofs that are quadratic in the size of the circuit implementing the NP relation.

In this work, we explore the possibility of building protocols that lie at the intersection of all of these desirable qualities. Specifically we address the following question:

> Can we construct constant-round UC-secure ZK protocols that are secure against adaptive corruptions, with proof size linear in the size of the circuit that implements the NP relation?

1.3 Our Contributions

Inspired by the recent progress in the domain of garbling schemes as primitives and interesting applications of garbled circuit (GC) based ZK protocols, we revisit ZK protocols from GCs. Recent works including [CGM16,KKL+16] make non-blackbox use of the GC-based ZK protocols of [JKO13], exploiting particularly the way the keys for the underlying GCs are constructed and the method by which the garbled input (i.e. witness) is encoded. Such applications will directly benefit from any improvement in the domain of garbled circuit based ZK protocols. Our contributions are listed below.

Efficient Constant-round Adaptively Secure ZK Protocols. While security against static adversaries provides a convenient stepping-stone for designing protocols against strong malicious attacks, a general real-life scenario certainly calls for adaptive security where the adversary can use its resources in a gradual fashion, making dynamic corruption decisions as the protocol progresses. Our first contribution is to show that the ZK protocol of [JKO13] can be proven to be adaptively secure in the UC setting if the underlying oblivious transfer (OT) primitive satisfies a mild adaptive security guarantee. Namely, we require that the receiver's communication can be equivocated to any input of the receiver. Such an OT is referred to as receiver equivocal OT (**RE-OT**). We show that the framework of [PVW08] itself, in one of its incarnation, provides **RE-OT**. Specifically, the mode of [PVW08] that offers statistical security for the receiver also offers the flavor of adaptive security that we demand from **RE-OT**. The main observation instrumental in crafting the adaptive proof of security for ZKGC is that the constructor of GC has no input. Therefore, the primary challenge of explaining the randomness of the GC construction in post-execution corruption case is bypassed.

Next, we focus on reducing the exact round complexity of ZKGC style protocols. We propose a three-round protocol. Since neither zero-knowledge proofs nor arguments can be achieved in less than four rounds without additional assumptions [GK96], we devise our protocols in the CRS model where the CRS is short unlike those used in SNARKs. Starting with ZKGC, our three-round protocol cuts down two rounds in [JKO13] using the idea of conditional opening [BP12] of a secret information that enables garbled circuit verification. That is, the key to GC verification can be unlocked only when the prover possesses a valid witness. Though fairly simple, implementing this idea makes the security proof of the resulting protocol challenging and subtle due to a circularity issue. Loosely speaking, when the prover does not hold a valid witness, the authenticity of GC should translate to the security of the key and at the same time, the security of the key should translate to the authenticity of the GC. We handle this issue by implementing the conditional disclosure via encryption in the Random Oracle Model (ROM). While the ZKGC protocol requires at least 5 rounds in its most round-efficient instantiation, we improve the complexity to three at *no* additional cost of communication (in fact with slight improvement), and little change in computation (one hash invocation versus a commitment in [JKO13]). We show this protocol to be adaptively secure too, when plugged in with **RE-OT**s.

In terms of concrete proof size (communication), our three-round protocol yields a better result than ZKBoo [GMO16] (and even it's more efficient successor ZKB++ [CDG$^+$17]) both in its interactive and non-interactive form with the right choice of the security parameters. We assume that circuit C computes the statement to be proven. While our three-round ZK needs a communication of $\lambda|C|$ bits (ignoring the circuit-independent parts), [GMO16] needs at least $3.41\lambda|C|$ to achieve the same $(\frac{1}{2^\lambda})$ soundness. In the table below, we compare our protocol asymptotically with the existing efficient constructions. Let 'PKE' and 'SKE' denote the number of public key and respectively secret key operations. We note that **RE-OT** can be efficiently constructed assuming DDH assumption, with no overhead over the regular OT in the framework of [PVW08].

Protocols	Proof Size	Prover Runtime	Verifier Runtime	Rounds	Assumptions	Security
ZKGC [JKO13]	$\mathcal{O}(\lambda \cdot \vert C\vert)$	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	5	Standard (OWF) +OT	Static (UC)
ZKBoo [GMO16]	$\mathcal{O}(\lambda \cdot \vert C\vert)$	$\mathcal{O}(\lambda\vert C\vert)$ SKE	$\mathcal{O}(\lambda\vert C\vert)$ SKE	1	PROM	Adaptive
ZKB++ [CDG$^+$17]	$\mathcal{O}(\lambda \cdot \vert C\vert)$	$\mathcal{O}(\lambda\vert C\vert)$ SKE	$\mathcal{O}(\lambda\vert C\vert)$ SKE	1	PROM	Adaptive
Ligero (Arithmetic)	$\mathcal{O}(\lambda^{1.5}\sqrt{\vert C\vert})$	$\mathcal{O}(\vert C\vert \log \vert C\vert)$ SKE	$\mathcal{O}(\vert C\vert \log \vert C\vert)$ SKE	1	PROM	Adaptive
Ligero (Boolean)	$\mathcal{O}(\lambda\sqrt{\vert C\vert \log \vert C\vert})$	$\mathcal{O}(\vert C\vert \log \vert C\vert)$ SKE	$\mathcal{O}(\vert C\vert \log \vert C\vert)$ SKE	1	PROM	Adaptive
[HV16]	$\mathcal{O}(\mu\vert C\vert\mathsf{poly}(\lambda))$	$\mathcal{O}(\mu\vert C\vert\mathsf{poly}(\lambda))$ SKE	$\mathcal{O}(\mu\vert C\vert\mathsf{poly}(\lambda))$ SKE	$\mathcal{O}(\mu \log \mu)$	Standard (OWP)	Adaptive
ZKGC **(This paper)**	$\mathcal{O}(\lambda \cdot \vert C\vert)$	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	5	Standard (OWF) + RE-OT (DDH)	Adaptive (UC)
This paper	$\mathcal{O}(\lambda \cdot \vert C\vert)$	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	$\mathcal{O}(\vert C\vert)$ SKE + $\mathcal{O}(n)$ PKE	3	ROM + RE-OT (DDH)	Adaptive (UC)

Table 1: Comparison among Zero Knowledge Protocols

2-Round Zero-knowledge Proofs. We next investigate the possibility of building efficient GC based ZK protocols with fewer than three rounds of interaction. In the spirit similar to that of [HV16], our two round protocol borrows techniques from non-interactive two-party computation (2PC) literature

[IKO+11,AMPR14,MR17] except for the following: We do not need the gadgets for input consistency checks of the prover, and input recovery mechanisms in case of inconsistent outputs [Lin13,LR14,RR16,MR17]. Our protocol is a proof, while most known efficient ZK protocols and our three round protocol are only arguments. The two round ZK may be cast as a sigma protocol and by applying the Fiat-Shamir transform, one may obtain NIZK arguments in the random oracle model. Finally, we observe that for the 2-round and NIZK argument we do not rely on the authenticity property of the garbling scheme. However, more efficient garbled circuit constructions by giving up on authenticity is precluded by the result of [AIK10]. While the result of [AIK10] needs to encode a different circuit (the underlying circuit augmented with a MAC computation) to achieve authenticity using a private scheme, we show a similar result while encoding the *same* underlying circuit. Both results essentially show that any garbling scheme that satisfies privacy also has authenticity.

1.4 Organization

We begin by briefly discussing definitions and constructions required for this work in Section 2. In Section 3 we show that the ZK protocol of [JKO13] is adaptively secure. Section 4 presents our three-round ZK protocol from conditional disclosure. Section 5 discusses our 2-round ZK. We include our result on authenticity-free garbling in Section 6.

2 Preliminaries

Notation. We denote probabilistic polynomial time by PPT. Let λ be the security parameter. $[n]$ and $[m, n]$ for $n > m$ denote the sets $\{1, ..., n\}$ and $\{m, m+1, \ldots, n\}$ respectively. $|\boldsymbol{t}|$ denote the number of bits in a string $\boldsymbol{t}$. We use $||$ to denote concatenation of bit strings, and write $x \stackrel{R}{\leftarrow} \mathcal{X}$ to mean sampling a value x uniformly from the set $\mathcal{X}$. A function $f(\cdot)$ is said to be negligible if $\forall c \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ such that $\forall n \geq n_0$, $f(n) < n^{-c}$. Let S be an infinite set and $X = \{X_s\}_{s \in S}, Y = \{Y_s\}_{s \in S}$ be distribution ensembles. We say X and Y are computationally indistinguishable, denoted by $X \stackrel{c}{\equiv} Y$, if for any PPT distinguisher $\mathcal{D}$ and all sufficiently large $s \in S$, we have $|\Pr[\mathcal{D}(X_s) = 1] - \Pr[\mathcal{D}(Y_s) = 1]| < 1/p(|s|)$ for every polynomial $p(\cdot)$. In the following, we review few building blocks. The ZK and Oblivious Transfer (OT) functionality are recalled in Appendix B.1.

2.1 Garbled Circuits

The work of Bellare et al. [BHR12b] formalizes *Garbling Schemes* as a primitive for modular use in cryptographic protocols, by defining several notions of security, including *obliviousness*, *privacy* and *authenticity*, of which we are interested in the latter two. Informally, privacy aims to protect the privacy of encoded inputs, while authenticity captures the unforgeability of the output of a garbled circuit evaluation. Majority of the schemes in the literature, including the classical scheme of Yao [Yao86], satisfy the two aforementioned properties. Using the language of [BHR12b] for circuits; the circuit itself is a directed acyclic graph, where each gate g is indexed by its outgoing wire, and its left and right incoming wires $A(g)$ and $B(g)$ are numbered such that $g > B(g) > A(g)$. Also, a circuit output wire can not be an input wire to any gate. We denote the number of input wires, gates and output wires using n, q and m respectively in a circuit C.

At a high-level, a garbling scheme consists of the following algorithms: Gb takes a circuit as input and outputs a garbled circuit, encoding information, and decoding information. En takes an input x and encoding information and outputs a garbled input $\mathbf{X}$. Ev takes a garbled circuit and garbled input $\mathbf{X}$ and outputs a garbled output $\mathbf{Y}$. De takes a garbled output $\mathbf{Y}$ and decoding information and outputs a plain circuit-output (or an error, $\perp$). Finally, we use an additional verification algorithm in the garbling scheme that output 1 or 0 based certain validity checks performed on a triple $(C, \mathbf{C}, e)$. Formally, a *garbling scheme* is defined by a tuple of functions $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De}, \mathsf{Ve})$, described as follows:

- *Garble* algorithm $\mathsf{Gb}\left(1^\lambda, C\right)$: A randomized algorithm which takes as input the security parameter and a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and outputs a tuple of strings $(\mathbf{C}, e, d)$, where $\mathbf{C}$ is the garbled circuit, e denotes the input-wire labels, and d denotes the decoding information.
- *Encode* algorithm $\mathsf{En}\,(x, e)$: a deterministic algorithm that outputs the garbled input $\mathbf{X}$ corresponding to input x.
- *Evaluation* algorithm $\mathsf{Ev}\,(\mathbf{C}, \mathbf{X})$: A deterministic algorithm which evaluates garbled circuit $\mathbf{C}$ on garbled input $\mathbf{X}$, and outputs a garbled output $\mathbf{Y}$.
- *Decode* algorithm $\mathsf{De}\,(\mathbf{Y}, d)$: A deterministic algorithm that outputs the plaintext output corresponding to $\mathbf{Y}$ or $\perp$ signifying an error if the garbled output $\mathbf{Y}$ is invalid.
- *Verify* algorithm $\mathsf{Ve}\,(C, \mathbf{C}, e)$: A deterministic algorithm which takes as input a circuit $C : \{0,1\}^n \mapsto \{0,1\}^m$, a garbled circuit (possibly malicious) $\mathbf{C}$, encoding information e, and outputs 1 when $\mathbf{C}$ is a valid garbling of C, and 0 otherwise.

A garbling scheme may satisfy several properties such as *correctness, privacy, authenticity and notions of verifiability.* The definitions for correctness, privacy and authenticity are standard: correctness enforces that a correctly garbled circuit, when evaluated, outputs the correct output of the underlying circuit; privacy aims to protect the privacy of encoded inputs; authenticity enforces that the evaluator can only learn the output label that corresponds to the value of the function. We use two notions of verifiability. One of the notions enforces that the garbling of a circuit indeed implements the specified plaintext circuit C. This notion of verification is used in our two-round protocol, NIZK and also in the Yao-based 2PC protocols using cut-and-choose (where the check circuits are verified according to this notion) [Lin13,LR14,RR16,MR17]. The other notion of verifiability introduced in [JKO13] enforces that the garbled output corresponding to a given clear output can be extracted for a verified tuple $(C, \mathbf{C}, e)$. This is used in our three round protocol. For the sake of completeness, we give the definitions of these properties in Appendix A.

We are interested in a class of garbling schemes referred to as *projective* in [BHR12b]. When garbling a circuit $C : \{0,1\}^n \mapsto \{0,1\}^m$, a projective garbling scheme produces encoding information of the form $e = \left(k_i^0, k_i^1\right)_{i \in [n]}$, and the encoded input $\mathbf{X}$ corresponding to $x = (x_i)_{i \in [n]}$ can be interpreted as $\mathbf{X} = \mathsf{En}(x, e) = (k_i^{x_i})_{i \in [n]}$.

2.2 Hash Function and Random Oracle Model

We use a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ which we model as a random oracle. Namely, we prove the security of our protocol assuming that H implements a functionality $\mathcal{F}_{\mathrm{RAND}}$ which for different inputs x, returns uniform random output values from the range of $H(x)$. In the proof, we rely on observability of H i.e. the reduction can observe the queries made to the H by the distinguisher of certain two views. Note that the simulator does not observe queries to the random oracle.

3 Adaptive Security of [JKO13]

In this section, we show that the garbled circuit based ZKGC protocol is adaptively secure when instantiated with an OT that satisfies a special property of Receiver Equivocality. We formalize the notion of Receiver Equivocal Oblivious Transfer which is an OT primitive with mild adaptive security guarantees. Essentially, we require that the view of a receiver be reconstructable in the case of a post-execution corruption. A similar notion was introduced in [Bea96b]. We show that the OT framework of [PVW08] is already receiver equivocal when it is instantiated with statistical security against a corrupt sender ("decryption mode"). We then show that when the zero-knowledge protocol of [JKO13] is instantiated with **RE-OT**, it achieves *adaptive security* without any additional effort. Below, we formulate **RE-OT**, recall the construction of [JKO13] (the schematic diagram is given in Appendix C), describe the adaptive proof of security of [JKO13] and conclude with an instantiation of **RE-OT**.

Definition of RE-OT. An oblivious transfer protocol is said to be receiver equivocal if it is possible to produce the receiver's message in the protocol *without committing to a choice bit.* For this to be meaningful, we also require that it be possible to efficiently generate the local randomness which when combined with either choice bit would make an honest receiver output the same message. This is formalized by requiring the existence of a simulator $\mathcal{S}^{\mathsf{RE}}$ which can perform this task, in Definition 3.1.

Definition 3.1 ***(RE-OT)*** *Let* $\Pi_{\mathsf{OT}} = (\Pi^{\mathsf{S}}_{\mathsf{OT}}, \Pi^{\mathsf{R}}_{\mathsf{OT}})$ *be a 2-round OT protocol securely implementing the* $\mathcal{F}_{OT}$ *functionality in the CRS model where* S *and* R *run their respective algorithms as specified by* $\Pi^{\mathsf{S}}_{\mathsf{OT}}(\mathsf{crs}, a_0, a_1, m^{\mathsf{R}}; r^{\mathsf{S}})$ *and*

$\Pi^R_{\mathsf{OT}}(\mathsf{crs}, \sigma; r^R)$ *respectively. Here,* a_0, a_1 *are the sender's inputs,* σ *is the receiver's choice bit,* r^S, r^R *are the sender's and receiver's respective local randomness, and* m^R *is the receiver's message. Let* $(\mathsf{crs}, t) \leftarrow \mathsf{Setup}(1^n, \mu)$ *be the output of the setup functionality which takes the security parameter and a mode* $\mu \in \{0, 1\}$*, and* t *is the corresponding trapdoor which is accessible only to the simulator* $\mathcal{S}$*. Then* Π_{OT} *is an* ***RE-OT*** *if the following conditions hold:*

- *Indistinguishability of modes: The CRSs of the two modes are computationally indistinguishable,*

$$\mathsf{crs}_0 \stackrel{c}{\equiv} \mathsf{crs}_1 \, \forall \, (\mathsf{crs}_0, t_0) \leftarrow \mathsf{Setup}(1^n, 0), (\mathsf{crs}_1, t_1) \leftarrow \mathsf{Setup}(1^n, 1)$$

- $\mathcal{F}_{OT}$ *in mode* 0*:* $\forall \, \mathsf{crs} \leftarrow \mathsf{Setup}(1^n, 0)$, $\Pi_{\mathsf{OT}} = \left(\Pi^S_{\mathsf{OT}}(\mathsf{crs}, a_0, a_1, m^R; r^S), \Pi^R_{\mathsf{OT}}(\mathsf{crs}, \sigma; r^R)\right)$ *securely implements the* $\mathcal{F}_{OT}$ *functionality.*
- *Equivocation in mode* 1*: There exists an algorithm* $\mathcal{S}^{\mathsf{RE}}(\mathsf{crs}, t)$ *which outputs* $\left(m^R, r^R_0, r^R_1\right)$ *such that* $m^R = \Pi^R_{\mathsf{OT}}(\mathsf{crs}, 0; r^R_0) = \Pi^R_{\mathsf{OT}}(\mathsf{crs}, 1; r^R_1)$*, and* $r^R_0, r^R_1 \stackrel{s}{\approx} r^R$*,* $\forall \, \mathsf{crs} \leftarrow \mathsf{Setup}(1^n, 1)$*.*

On the use of a CRS. We note here that there is nothing inherent in receiver equivocation that demands a CRS to implement **RE-OT**. We are interested in achieving UC-security, and so as to allow the protocol of [PVW08] as an instantiation of our definition, we assume that the protocol realizing **RE-OT** will make use of a CRS. However, this does not preclude the existence of **RE-OT** in the standalone model without a CRS, or even a UC-secure **RE-OT** in the Global Random Oracle hybrid model [CJS14] alone.

3.1 Recap of [JKO13]

We recall the ZKGC protocol below in the $(\mathcal{F}_{\mathsf{COT}}, \mathcal{F}_{\mathsf{COM}})$ hybrid model (a schematic diagram is given in Appendix C). The functionalities are presented in Appendix B.1.

3.2 Proof of Adaptive Security for [JKO13] from RE-OT

In this section we show that instantiating the ZKGC protocol with **RE-OT** satisfying Definition 3.1 yields a UC-secure protocol realizing $\mathcal{F}^R_{\mathsf{ZK}}$ (see Figure 12) tolerating adaptive adversaries.

Recalling Static Proof of Security. The simulator for a corrupt P plays the role of an honest verifier V. It constructs and communicates a correct garbled circuit, extracts the witness acting on behalf of $\mathcal{F}_{\mathsf{COT}}$ functionality, and accepts the proof only if the extracted witness is a valid one. On the other hand the real verifier accepts when the opening of the commitment is the correct output wire key Z. In $\mathcal{F}_{\mathsf{COM}}$-hybrid model, we can show that a malicious prover who is able make a real verifier output 'accept' (but not the simulator) can be used to break

Π_{ZKGC}

- **Oracles and Cryptographic Primitives:** A *correct, authentic, verifiable* garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, (\mathsf{Ve}_1, \mathsf{Ve}_2))$ (according to Definition A.5). A committing OT oracle $\mathcal{F}_{\mathsf{COT}}$.
- **Common Inputs of P and V:** A security parameter λ, relation R realized by circuit C, statement z.
- **Input of P:** A witness x of size $n = \mathsf{poly}(\lambda)$ such that $R(z, x) = 1$.
- **Input of V:** Nothing.

Witness input phase: For all $i \in [n]$, P sends $(\texttt{choose}, id, x_i)$ to $\mathcal{F}_{\mathsf{COT}}$.

GC Construction and wire label transfer phase: V garbles the circuit, $\left(\mathbf{C}, \left(K_i^0, K_i^1\right)_{i \in [n]}, Z\right) \leftarrow \mathsf{Gb}\left(1^\kappa, C\right)^a$. On receiving messages $(\texttt{chosen}, id)$ for $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$, V sends $\left(\texttt{transfer}, id, K_i^0, K_i^1\right)$ as input to $\mathcal{F}_{\mathsf{COT}}$ for all $i \in [n]$.

GC Evaluation and output commitment phase: P receives $(\texttt{transferred}, id, K_i^{x_i})$ for $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$, and parses $X = K_1^{x_1} \cdots K_i^{x_i} \cdots K_n^{x_n}$. P obtains $Z' = \mathsf{Ev}(C, X)$ and sends $(\texttt{commit}, id, Z')$ to $\mathcal{F}_{\mathsf{COM}}$.

GC verification and conditional output disclosure phase: On receiving $(\texttt{committed}, id, |Z'|)$ from $\mathcal{F}_{\mathsf{COM}}$, V sends the message $(\texttt{open-all}, id)$ to $\mathcal{F}_{\mathsf{COT}}$. On receiving $\left(\texttt{transfer}, id, K_i^0, K_i^1\right)$ for all $i \in [n]$ from $\mathcal{F}_{\mathsf{COT}}$, P verifies if the garbled circuit C which sent by the verifier earlier was correctly constructed.

i if $\mathsf{Ve}\left(\mathbf{C}, f, \left\{K_i^0, K_i^1\right\}_{i \in [n']}\right) \neq 1$, P aborts.

ii else P sends $(\texttt{reveal}, id)$ to $\mathcal{F}_{\mathsf{COM}}$.

Final verification phase: On receiving the message $(\texttt{reveal}, id, Z')$ from $\mathcal{F}_{\mathsf{COM}}$, V outputs $\texttt{accept}$ if $Z = Z'$.

[a] Instead of returning d, Gb is tweaked to return the 1-key on the output wire.

Fig. 1: Zero-knowledge from Garbled Circuits [JKO13]

authenticity of the underlying garbling scheme. We can use such a malicious prover P^* to construct an adversary $\mathcal{A}$ for the authenticity game of [BHR12b] as follows:

1. $\mathcal{A}$ receives the invalid witness x^* from P^* on behalf of $\mathcal{F}_{\mathsf{COT}}$ and forwards it to the authenticity challenger.
2. $\mathcal{A}$ receives $\mathbf{C}, X$ from the authenticity challenger and forwards it to P^*
3. $\mathcal{A}$ receives forged key Z' from P^* on behalf of $\mathcal{F}_{\mathsf{COM}}$ and submits it to the authenticity challenger.

Clearly, the event that $\mathcal{A}$ successfully forges an output for the given $\mathbf{C}, X$ is equivalent to the event that P^* convinces a verifier to output 'accept' without a valid witness. By authenticity of the garbling scheme, this event occurs with negligible probability.

The simulator for a corrupt V receives the encoding information from V on behalf of the $\mathcal{F}_{\mathsf{COT}}$ functionality and extracts the the output 1-key Z using received garbled circuit and encoding information. It then sends Z to the verifier only after receiving the correct encoding information from V in the open-all phase. Otherwise, it sends $\perp$ to V. Security in this case follows from the verifiability (that allows extraction of the output key from encoding information) of the underlying garbling scheme.

Adaptive Proof of Security. The bottleneck faced in simulating garbled circuit based protocols for post-execution corruptions usually lies in "explaining" the randomness of the GC constructor once her input is known. In the case of two-party computation, equivocating the view of the garbled circuit constructor requires heavy machinery such as in Canetti et al. [CPV17]. However in the ZKGC protocol verifier V is the GC constructor and *has no input*. The simulator can therefore run the code of honest V, which includes being an honest sender in the OT protocol (this is also why our OT need not achieve full-fledged adaptive security). On the prover's side, receiver equivocality of the OT allows a simulator to equivocate an adaptively corrupted prover's view of the OT protocol, as per the witness once known. We make the observation that *every step of* P *following the OT is independent of the witness*. Specifically, once the output key Z has been obtained by evaluating the GC sent by V, P does not use the witness again. Note that the simulator does not need the witness to obtain Z; the ZKGC simulator invokes the Π_{OT} simulator in order to extract all inputs of V and obtain all keys of the GC. Once the simulator obtains Z, the code of honest P can be run to complete the simulation. The implication of this for simulation of a post-execution corruption of P is that no additional work needs to be done besides equivocating the view of P in the OT. We now give a formal proof for all the cases:

- **Simulation for** V**.** The verifier, until it is corrupted, can be simulated following the static simulator for the corrupt P, irrespective of when P is corrupted. As recalled above, the simulation can be carried out by running the code of honest verifier (constructing a correct garbled circuit, participating in the **RE-OT**s with the correct encoding information and sending the correctly constructed garbled circuit). Upon corruption, the simulator can explain to the corrupt V the communication by means of the randomness used in its honest execution of V's code. The indistinguishability follows from the proof in the static corrupt prover case.
- **Simulation for** P**.** If the prover is corrupted at the outset, then the crs is set in mode 0. Otherwise, we consider the worst scenario of post-execution corruption, and set the crs in mode 1. If the verifier is also not corrupt during the construction of the garbled circuit, then simulator acts on behalf of both the honest parties and runs the code of honest verifier. In the $\mathcal{F}_{\mathsf{COM}}$-hybrid model, the simulator, without having access to the actual witness, runs $\left(m^{\mathsf{R}}, r_0^{\mathsf{R}}, r_1^{\mathsf{R}}\right) \leftarrow \mathcal{S}^{\mathsf{RE}}(\mathsf{crs}, t)$ to generate the transcript that needs to be communicated on behalf of P in **RE-OT** instances. The rest of the simulation is straight-forward irrespective of whether the verifier is corrupt or not. In the final step, the simulator may have to communicate Z which it picked itself while simulating V in this case. When P is corrupt in the end, its input

x_i to the i^{th} **RE-OT** instance can be explained as per any input using the randomness $r^{\mathsf{R}}_{x_i}$ returned by $\mathcal{S}^{\mathsf{RE}}$ of the **RE-OT**s. On the other hand, if V was corrupt before the garbled circuit construction phase, then the simulator gets Z via unlocking the GC using encoding information extracted from the corrupt V's communication. The rest remains the same as the previous case. Security in the former case follows via receiver equivocality of **RE-OT**. In the latter, it follows additionally from verifiability that ensures the encoding information leads to the correct Z with high probability.

3.3 Instantiation of RE-OT

The OT framework of [PVW08] is already receiver equivocal as per Definition 3.1. The protocol can be constructed efficiently under the Decisional Diffie Hellman, Quadratic Residuosity, or Learning With Errors hardness assumptions. Recall that the constructions of [PVW08] operate in two modes: messy and decryption, that corresponds to mode 0 and 1 respectively of our definition. The construction satisfies Definition 3.1 when instantiated in "decryption mode". In the simulation, when the receiver is corrupted before the first message is sent, the simulator sets the CRS in the messy mode, and no equivocation is necessary. Otherwise, the simulator sets the CRS in the decryption mode. Here we recall the instantiation of Π_{PVW} under the DDH hardness assumption and describe $\mathcal{S}^{\mathsf{RE}}_{\mathsf{PVW}}$ in the decryption mode. (Fig. 2).

Theorem 3.2 *The protocol Π_{PVW} in Fig. 2 is a **RE-OT**, assuming that DDH is hard for $\mathbb{G}$.*

Proof. The protocol Π_{PVW} in Fig. 2 is proven to realize the $\mathcal{F}_{\mathsf{OT}}$ functionality in the UC model by Peikert et al. [PVW08]. It is easy to see how $\mathcal{S}^{\mathsf{RE}}_{\mathsf{PVW}}$ allows for receiver equivocation as per Def. 3.1 when the crs is generated in mode 1:

- The randomness r^{R}_{σ} provided is interpreted as R's secret exponent α.
- Recall that the message m^{R} is (g_0^r, h_0^r), and candidate randomness output by $\mathcal{S}^{\mathsf{RE}}_{\mathsf{PVW}}$ is $r^{\mathsf{R}}_0 = r$, and $r^{\mathsf{R}}_1 = r^{\mathsf{R}}_0 \cdot t^{-1} = r \cdot t^{-1}$
- Correctness of message m^{R} can be seen as follows:
  1. $\Pi_{\mathsf{PVW}}\left(\mathsf{crs}, 0; r^{\mathsf{R}}_0\right)$ will output $\left(g_0^{r^{\mathsf{R}}_0}, h_0^{r^{\mathsf{R}}_0}\right) = (g_0^r, h_0^r) = m^{\mathsf{R}}$
  2. $\Pi_{\mathsf{PVW}}\left(\mathsf{crs}, 1; r^{\mathsf{R}}_1\right)$ will output $\left(g_1^{r^{\mathsf{R}}_1}, h_1^{r^{\mathsf{R}}_1}\right) = \left(g_1^{(r\cdot t^{-1})}, h_1^{(r\cdot t^{-1})}\right)$

     Recall that the trapdoor t relates g_0 to g_1 as $g_0^t = g_1$ and similarly $h_0^t = h_1$. Therefore we have that $\left(g_1^{(r\cdot t^{-1})}, h_1^{(r\cdot t^{-1})}\right) = (g_0^r, h_0^r) = m^{\mathsf{R}}$
- Finally, $r^{\mathsf{R}}_0, r^{\mathsf{R}}_1 = r, r \cdot (t^{-1})$ are clearly uniformly random, as r is sampled uniformly at random.

□

Also note that **RE-OT** is strictly weaker than OT with security against adaptive corruptions; any protocol satisfying the latter notion will necessarily be receiver-equivocal in order for the receiver's view to be fully simulatable in the event of a post-execution corruption.

Π_{PVW}

The parties have access to a common reference string $\mathsf{crs} \in \mathbb{G}^4$. Operations are over group $\mathbb{G}$ of prime order q, generated by g.

$\mathsf{Setup}(1^n, 0)$:
$\mathsf{crs} = (g_0, h_0, g_1, h_1) \in \mathbb{G}^4$. The trapdoor available to the simulator is $t = (t_0, t_1)$ such that $g_0^{t_0} = h_0$ and $g_1^{t_1} = h_1$.

$\mathsf{Setup}(1^n, 1)$:
$\mathsf{crs} = (g_0, h_0, g_1, h_1) \in \mathbb{G}^4$. The trapdoor available to the simulator is t such that $g_0^t = g_1$ and $h_0^t = h_1$.

$\Pi^{\mathsf{R}}_{\mathsf{PVW}}(\mathsf{crs}, \sigma)$:

- Sample $\alpha \in \mathbb{Z}_q$ uniformly at random.
- Compute $g = (g_\sigma)^\alpha$, $h = (h_\sigma)^\alpha$
- Send (g, h)

$\Pi^{\mathsf{S}}_{\mathsf{PVW}}\left(\mathsf{crs}, a_0, a_1, m^{\mathsf{R}}\right)$:

- Sample random elements r_0, s_0, r_1, s_1 from $\mathbb{Z}_q$.
- Compute $u_0 = g_0^{r_0} h_0^{s_0}$, $v_0 = g^{r_0} h^{s_0}$, $u_1 = g_1^{r_1} h_1^{s_1}$, $v_1 = g^{r_1} h^{s_1}$.
- Send $(u_0, w_0 = v_0 a_0)$, $(u_1, w_1 = v_1 a_1)$

R can retrieve the chosen message as $a_\sigma = w_\sigma \cdot (u_\sigma)^{-\alpha}$

$\mathcal{S}^{\mathsf{RE}}(\mathsf{crs}, t)$:

- Sample $r \in \mathbb{Z}_q$ and compute $m^{\mathsf{R}} = (g_0^r, h_0^r)$.
- Compute local randomness for both possible receiver inputs as $r_0^{\mathsf{R}} = r$ and $r_1^{\mathsf{R}} = r \cdot t^{-1}$.
- Output $(m^{\mathsf{R}}, r_0^{\mathsf{R}}, r_1^{\mathsf{R}})$

Fig. 2: **RE-OT** assuming DDH: as per [PVW08]

4 Zero Knowledge in Three Rounds

In this section, we present a *3-round ZK protocol* against a malicious verifier requiring just one GC in the non-programmable random oracle model, with *no increase in communication complexity*. Our protocol achieves this by a technique for non-interactive GC verification which allows us to remove the commitment and OT-open-all phases from ZKGC. Our approach is reminiscent of the technique of *conditional disclosure of secrets* (CDS)[GIKM98]. CDS has since been generalized [IW14], and used in several works, including in applications to improve round complexity of protocols [AIR01,BCPW15]. We show that the protocol is adaptively secure when the underlying OTs are receiver equivocal.

4.1 High-Level Idea

The high round cost of ZKGC makes it undesirable for many applications. However its usage of only one GC for an actively secure protocol is an attractive feature, prompting us to examine whether we can improve on the number of rounds required to realize ZK with only one GC. We now describe our intuition behind the protocol, beginning with informal observations about the number of rounds in ZKGC. Assuming the ZKGC paradigm to be broadly characterized by a protocol where the verifier V constructs a GC which is then evaluated by prover P, we make the following (informal) observations:

- As V constructs the GC, P's witness bits must be encoded as garbled input and delivered by means of an OT. The most efficient UC-secure OT in the literature [PVW08] requires 2 rounds to instantiate.
- Assuming the underlying GC to be statically secure in the terminology of Bellare et al. [BHR12a], the GC can at best be sent to P along with the final message of the OT (if not after the OT).
- P must communicate some information as a 'response' to V's GC 'challenge'; for instance the garbled output obtained as a result of evaluating the GC with her witness. This must necessarily be after she receives the GC, adding at least one more round after the OT.

In summary, it appears that the ZKGC paradigm requires at least 2 rounds for the OT, plus the GC transmission, and one round following that. Therefore, a 3-round ZK protocol appears to be optimal in the ZKGC paradigm, informally suggesting the optimality of our protocol. In the following, we make several observations that are instrumental to our protocol.

Conditional Verification of Garbled Circuits. We begin by making the following observation about the original ZKGC protocol: even a prover who does not have a witness is given the chance to first commit to her garbled output and verify that the GC she received was correctly generated. Verification of the GC is a process that takes two additional rounds of interaction in their protocol. We ask, can we use conditional disclosure of secrets to reduce the number of rounds: *"can we provide some additional information with a GC that will allow an evaluator to non-interactively verify that the GC was correctly constructed only when it possess a valid witness?"* We answer this question in the affirmative, at least for the ZKGC setting. An idea somewhat similar in spirit was proposed in [BP12] to construct a three-round 'weak' ZK protocol from a garbling scheme and point-obfuscation. That is, knowing the witness gives the prover access to a secret via a garbled circuit handed over by the verifier. The secret, then, can be used to unlock the seed that opens the garbled circuit and enables verifying the correct construction of the GC. Technique-wise, we depart from the work of [BP12] as follows. The secret is encoded in the circuit output in [BP12] and hence, privacy of the garbling circuit is one of the properties they rely on to achieve soundness. On the contrary, the secret, in our case is the output key corresponding to bit 1 and hence, soundness is achieved via authenticity.

Qualitatively, their protocol is not a full-fledged ZK, is in the plain model, has a non-black-box simulator and relies on strong assumptions such as obfuscation. Our ZK protocol is proven UC-secure with a black-box simulator and relies on standard assumptions, albeit assuming a CRS setup.

Interestingly, the intuition behind the ability of [JKO13] to achieve full black-box simulation was that the relaxation in round complexity rendered the four-round barrier in the plain model [GK96] inapplicable. However, our result demonstrates that the trusted setup required to implement a full black-box simulatable two-round OT is sufficient to construct a three round zero-knowledge argument using the concretely efficient [JKO13] technique and a non-programmable random oracle.

Our intuition is implemented as follows: Given that $\left(\mathbf{C}, \left\{(k_j^0, k_j^1)\right\}_{j\in[n]}, (k^0, k^1)\right) \leftarrow \mathsf{Gb}\left(1^\lambda, C\right)$ and an honest P has obtained encoded input $\mathbf{X} = \left(k_j^{x_j}\right)_{j\in[n]}$ for a witness $x = (x_1 \ldots, x_n)$, she can compute $k^1 = \mathsf{Ev}\left(\mathbf{C}, \mathbf{X}\right)$. Now that P has evaluated the GC, we wish to enable her to 'open' the GC and verify that it was constructed correctly. To do this, we provide her with a ciphertext encrypting some useful information. Concretely, the ciphertext $T = H(k^1) \oplus r^{\mathsf{S}}$, where H is a random oracle and r^{S} contains the randomness used by the sender in the OT instances. Once P gets this randomness, she can unlock $\left\{k_j^0, k_j^1\right\}_{j\in[n]}$ and can verify if the circuit has been constructed correctly. In the following, we formalize the property needed from the OT protocol, namely that the randomness of the sender reveals the inputs of the sender.

Sender-Extractability of OT. Let $\Pi_{\mathsf{OT}} = (\Pi_{\mathsf{OT}}^{\mathsf{S}}, \Pi_{\mathsf{OT}}^{\mathsf{R}})$ be a 2-round OT protocol securely implementing the $\mathcal{F}_{\mathsf{OT}}$ functionality in the CRS model where S and R run their respective algorithm as specified by $\Pi_{\mathsf{OT}}^{\mathsf{S}}$ and $\Pi_{\mathsf{OT}}^{\mathsf{R}}$ respectively. Let crs be the string that both parties have access to. We denote the first message of the protocol sent by the receiver R by $m^{\mathsf{R}} = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, \sigma; r^{\mathsf{R}})$ where σ is R's choice bit and r^{R} his randomness. Let the input of the sender S be a_0, a_1; we denote the second message of the OT protocol, sent by S, by $m^{\mathsf{S}} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, a_0, a_1, m^{\mathsf{R}}; r^{\mathsf{S}})$. The receiver can now compute the chosen message, $x_\sigma = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, \sigma, m_{\mathsf{S}}; r^{\mathsf{R}})$. We assume that Π_{OT} has the following sender-extractable property: revealing the randomness of the sender, allows the receiver to reconstruct the sender's messages correctly with high probability. That is, there exists a public efficiently computable function, Ext such that $\mathsf{Ext}(\mathsf{crs}, \mathcal{T}_{\mathsf{OT}}(a_0, a_1, \sigma), r^{\mathsf{S}})$ outputs (a_0, a_1) where $\mathcal{T}_{\mathsf{OT}}(a_0, a_1, \sigma)$ refers to the transcript of Π_{OT} with sender's input as a_0, a_1 and receiver's input as σ. Namely, $\mathcal{T}_{\mathsf{OT}}(a_0, a_1, \sigma) = (m^{\mathsf{R}}, m^{\mathsf{S}})$ where m^{R} and m^{S} are as defined above.

Definition 4.1 *A protocol Π_{OT} is a secure sender-extractable OT protocol if*

– *it securely implements $\mathcal{F}_{\mathsf{OT}}$ in the presence of malicious adversaries, and*
– $\forall\ a_0, a_1, \sigma,$ *such that* $|a_0|, |a_1| \leq poly\,(\lambda)$, $\sigma \in \{0, 1\}$, $\exists$ *a PPT algorithm* Ext *such that the following probability is negligible in* λ.

$$\Pr\left((a_0', a_1') \neq (a_0, a_1) : \mathsf{Ext}(\mathsf{crs}, \mathcal{T}_{\mathsf{OT}}(a_0, a_1, \sigma), r^{\mathsf{S}}) = (a_0', a_1')\right).$$

We note that the protocol of [PVW08] is UC-secure in the CRS model, is 2-rounds, and satisfies the sender-extractability property of Definition 4.1. We use such a protocol in our construction.

4.2 Our Construction

At a high-level, our construction proceeds as follows. The verifier constructs a garbled circuit of the circuit C implementing the relation. The prover obtains the wire keys corresponding to his witness via an OT protocol. Now, the verifier sends the garbled circuit to the prover, and, in addition, a ciphertext. This ciphertext allows the prover to open and verify the garbled circuit, but only if he possesses a valid witness. The complete description of our protocol Π_{ZK3} is presented in Figure 3. A schematic diagram of the protocol idea is given in Appendix C (Figure 16). We now prove security of Π_{ZK3} in Universal Composability (UC) framework recalled briefly in Appendix B. As we do not rely on programming the Random Oracle, we can also adapt our proof in the UC setting to use a Global Random Oracle [CJS14].

Theorem 4.2 *Let* Garble *be a correct, authentic, verifiable (according to Definition A.5) garbling scheme,* Π_{OT} *be an sender-extractable OT protocol, and* H *be an extractable random oracle. The protocol* Π_{ZK3} *in Figure 3 securely implements* $\mathcal{F}_{\mathsf{ZK}}^{R}$ *in the presence of malicious adversaries.*

Proof. To prove the security of our protocol, we describe two simulators. The simulator $\mathcal{S}_{\mathsf{P}}$ simulates the view of a corrupt prover and appears in Fig. 4. The simulator $\mathcal{S}_{\mathsf{V}}$ simulates the view of a corrupt verifier and is presented in Fig. 5.

Security against a Corrupt Prover $\mathsf{P}^\star$. We now prove that $\textsc{ideal}_{\mathcal{F}_{\mathsf{ZK}}^{R},\mathcal{S}_{\mathsf{P}},\mathcal{Z}} \overset{c}{\approx} \textsc{real}_{\Pi_{\mathsf{ZK3}},\mathcal{A},\mathcal{Z}}$ when $\mathcal{A}$ corrupts P. We begin by noting that the simulated and the real worlds are identical when P uses a valid witness x. The view of a malicious $\mathsf{P}^\star$ who does not possess a valid witness x is proven to be computationally close to the simulation through an intermediate hybrid $\textsc{hyb}_1$. The hybrid $\textsc{hyb}_1$ is constructed identically to $\textsc{ideal}_{\mathcal{F}_{\mathsf{ZK}}^{R},\mathcal{S}_{\mathsf{P}},\mathcal{Z}}$ with the exception of the criterion to output accept. In $\textsc{hyb}_1$, the verifier accepts if $\mathsf{P}^\star$ outputs the correct k^1 (as in the $\textsc{real}$ view) regardless of the witness used. We begin our analysis by noting that unless a $\mathsf{P}^\star$ queries the correct k^1 to the random oracle H, the string T appears completely random. Therefore, given that a $\mathsf{P}^\star$ attempting to distinguish between the $\textsc{real}$ view and the view generated by $\textsc{hyb}_1$, we branch our analysis into the following cases:

- $\mathsf{P}^\star$ **does not output the correct k^1 in either world.** Here we assume that a $\mathsf{P}^\star$ also does not query the correct k^1 to the random oracle H to be able to unlock ciphertext T. If the prover does indeed query the correct k^1 to H with non-negligible probability, we move on to the next case. A $\mathsf{P}^\star$ who is successful in distinguishing $\textsc{real}_{\Pi_{\mathsf{ZK3}},\mathcal{A},\mathcal{Z}}$ from $\textsc{hyb}_1$ in this case can be used to break OT sender security. The reduction computes a garbled circuit $\mathbf{C}$ and sends the input keys to the OT challenger (by means of the environment

Π_{ZK3}

- **Oracles and Cryptographic Primitives:** A *correct, authentic, verifiable* garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, (\mathsf{Ve}_1, \mathsf{Ve}_2))$ (according to Definition A.5). A sender-extractable 2-round OT Π_{OT} with the common reference string crs. A hash function $H : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\lambda)}$ which we model as a random oracle.
- **Common Inputs of P and V:** A security parameter λ, relation R realized by circuit C, statement z, common reference string crs for Π_{OT}.
- **Input of P:** A witness x of size $n = \mathsf{poly}(\lambda)$ such that $R(z, x) = 1$.
- **Input of V:** Nothing.

OT First Message Phase: P plays the role of the receiver R in n instances of Π_{OT} and provides his witness bit x_j as input to the jth instance of Π_{OT}. Specifically, it:
- chooses $r_j^{\mathsf{R}} \xleftarrow{R} \{0,1\}^\lambda$, and computes $m_j^{\mathsf{R}} = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, x_j; r_j^{\mathsf{R}}), \forall j \in [n]$ as the first message in the jth instance of Π_{OT}
- sends $\{m_j^{\mathsf{R}}\}_{j\in[n]}$ to V.

GC Construction and OT Second Message Phase: V constructs a garbled circuit $\mathbf{C}$ for C as $(\mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}, (k^0, k^1)) \leftarrow \mathsf{Gb}(1^\lambda, C)$. V now provides the wire labels for the input wires of $\mathbf{C}$ by playing the role of the sender S in n instances of Π_{OT}. Specifically, it
- samples randomness $r_j^{\mathsf{S}} \xleftarrow{R} \{0,1\}^\lambda$, $\forall j \in [n]$ and parses $r^{\mathsf{S}} = r_1^{\mathsf{S}} || \cdots || r_n^{\mathsf{S}}$
- computes $m_j^{\mathsf{S}} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, k_j^0, k_j^1, m_j^{\mathsf{R}}; r_j^{\mathsf{S}}), \forall j \in [n]$ and $T = H\left(k^1\right) \oplus r^{\mathsf{S}}$ and
- sends $(\mathbf{C}, \{m_j^{\mathsf{S}}\}_{j\in[n]}, T)$ to P.

P computes the wire-keys corresponding to his input: $k_j^{x_j} = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, m_j^{\mathsf{R}}, m_j^{\mathsf{S}}, r_j^{\mathsf{R}}), \forall j \in [n]$.

GC Evaluation, Verification and Output Disclosure Phase: P evaluates $\mathbf{C}$ and obtains the garbled output. He then recovers the randomness used by the sender (namely, V) using the output-wire key he obtained. By the sender-extractability of Π_{OT}, P recovers the input-wire labels which are the OT inputs of V. P can now verify that the garbled circuit was correctly constructed using the recovered wire keys. Specifically, it:
- executes $\mathbf{Y} = \mathsf{Ev}(\mathbf{C}, \{k_j^{x_j}\}_{j\in[n]})$
- recovers $r^{\mathsf{S}} = H(\mathbf{Y}) \oplus T$, and parses $r^{\mathsf{S}} = r_1^{\mathsf{S}} || \cdots || r_n^{\mathsf{S}}$
- aborts if $\exists j$ such that $\mathsf{Ext}(\mathsf{crs}, m_j^{\mathsf{R}}, m_j^{\mathsf{S}}, r_j^{\mathsf{S}}) = \bot$ and extracts $(k_j^0, k_j^1) = \mathsf{Ext}(\mathsf{crs}, m_j^{\mathsf{R}}, m_j^{\mathsf{S}}, r_j^{\mathsf{S}}), \forall j \in [n]$ otherwise
- aborts if $\mathsf{Ve}_2(C, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 0$ and sends $\mathbf{Y}$ to V otherwise.

Output Phase: If $\mathbf{Y} = k^1$, then V outputs `accept`, else he outputs `reject`.

Fig. 3: 3-round GC based Zero Knowledge protocol

for the OTs) as the sender's input. The reduction then extracts the input x of $\mathsf{P}^\star$ and forwards to the OT challenger as the choice bits of the receiver. The response of OT challenger who computes the sender's message either by invoking a real sender i.e. as $m_j^{\mathsf{S}} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, k_j^0, k_j^1, m_j^{\mathsf{R}}; r_j^{\mathsf{S}}), \forall j \in [n]$ or by invoking a simulator i.e. as $m_j^{\mathsf{S}} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, k_j^{x_j}, 0^\lambda, m_j^{\mathsf{R}}; r_j^{\mathsf{S}}), \forall j \in [n]$ is sent to the reduction who further forwards the message to $\mathsf{P}^\star$ along with $\mathbf{C}$ and a random T. In case the OT challenger invokes a simulator the view of $\mathsf{P}^\star$ is

Simulator $\mathcal{S}_{\mathsf{P}}$

The simulator plays the role of the honest V and simulates each step of the protocol Π_{ZK3} as follows. The communication of the $\mathcal{Z}$ with the adversary $\mathcal{A}$ who corrupts P is handled as follows: Every input value received by the simulator from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to the simulator's output tape (to be read by the environment $\mathcal{Z}$).

OT First Message Phase: $\mathcal{S}_{\mathsf{P}}$ invokes the simulator of Π_{OT} for corrupt receiver and extracts P's input bit to the jth instance of Π_{OT}, namely the jth witness bit x_j.

GC Construction and OT Second Message Phase: $\mathcal{S}_{\mathsf{P}}$ emulates an honest V if the extracted witness x is valid i.e. $R(z, x) = 1$. Otherwise, $\mathcal{S}_{\mathsf{P}}$ does the following:
- It constructs a garbled circuit $\mathbf{C}$ for C as $(\mathbf{C}, \{(k_j^0, k_j^1)\}_{j\in[n]}, (k^0, k^1)) \leftarrow \mathsf{Gb}(1^\lambda, C)$.
- It samples r^{S} uniformly at random and parses it as $r^{\mathsf{S}} = r_1^{\mathsf{S}}||\cdots||r_n^{\mathsf{S}}$,
- It computes $m_j^{\mathsf{S}} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, k_j^{x_j}, 0^\lambda, m_j^{\mathsf{R}}; r_j^{\mathsf{S}}), \forall j \in [n]$ and samples T uniformly at random and
- It sends $(\mathbf{C}, \{m_j^{\mathsf{S}}\}_{j\in[n]}, T)$ to $\mathsf{P}^\star$.

GC Evaluation, Verification and Output Disclosure Phase: $\mathcal{S}_{\mathsf{P}}$ does nothing in this step.

Output Phase: $\mathcal{S}_{\mathsf{P}}$ sends x to $\mathcal{F}_{\mathsf{ZK}}^R$ on behalf of $\mathsf{P}^\star$ if $R(z, x) = 1$. Otherwise, it sends $\perp$.

Fig. 4: Simulator $\mathcal{S}_{\mathsf{P}}$

identical to $\textsc{hyb}_1$, whereas when the OT challenger uses a real execution of Π_{OT} the view of $\mathsf{P}^\star$ is identical to $\textsc{real}$ (T is random given that the correct k^1 is never queried to H). Therefore, the probability of distinguishing between the REAL and $\textsc{hyb}_1$ view translates to the probability of distinguishing between the real and the simulated view of the OT protocols for the case when the receiver is corrupt.

- **$\mathsf{P}^\star$ outputs the correct k^1 in $\mathbf{real}_{\Pi_{\mathsf{ZK3}},\mathcal{A},\mathcal{Z}}$ with significantly higher probability than in $\mathbf{hyb}_1$.** This case is similar to the previous case in that $\mathsf{P}^\star$ can be used to break sender security of the OT by computing $\mathbf{C}$ locally in the reduction. If $\mathsf{P}^\star$ outputs a correct k^1, the reduction is interacting with Π_{OT} whereas if not, the challenger must have invoked the simulator for Π_{OT}. The advantage of this reduction is the difference in probabilities with which $\mathsf{P}^\star$ forges k^1 successfully in the $\textsc{real}$ and $\textsc{hyb}_1$ worlds.
- **$\mathsf{P}^\star$ outputs the correct k^1 in both worlds with almost the same probability.** The corrupt $\mathsf{P}^\star$ can be used directly to break authenticity of the garbling scheme. Clearly the OT message corresponding to inactive input keys are not used by the corrupt P; the ability to output the correct k^1 must be derivative of the ability to forge a key for the garbled circuit alone. It is therefore straightforward to use $\mathsf{P}^\star$ to forge k^1 for a given garbled circuit $\mathbf{C}$, as its view can be generated as per $\textsc{hyb}_1$, which does not require the inactive garbled circuit keys to compute the OT messages.

Note that in Cases 2 and 3, we consider a $\mathsf{P}^\star$ who outputs k^1 to be equivalent to a $\mathsf{P}^\star$ who queries the random oracle on k^1 to unlock T in its effort to distinguish REAL from HYB_1. Instead of receiving k^1 directly from $\mathsf{P}^\star$, our reductions will observe its query to the random oracle.

Finally $\text{IDEAL}_{\mathcal{F}_{\mathsf{ZK}}^R,\mathcal{S}_\mathsf{P},\mathcal{Z}}$ deviates from HYB_1 only in its criteria to output accept. Only a corrupt P who is able to output k^1 will be able to distinguish HYB_1 from $\text{IDEAL}_{\mathcal{F}_{\mathsf{ZK}}^R,\mathcal{S}_\mathsf{P},\mathcal{Z}}$. Such a P can be used directly to forge an output key for a given $\mathbf{C}$ with the same probability (which by authenticity of the garbling scheme, must be negligible).

Simulator $\mathcal{S}_\mathsf{V}$

The simulator plays the role of the honest P and simulates each step of the protocol Π_{ZK3} as follows. The communication of the $\mathcal{Z}$ with the adversary $\mathcal{A}$ who corrupts V is handled as follows: Every input value received by the simulator from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to the simulator's output tape (to be read by the environment $\mathcal{Z}$).

OT First Message Phase: $\mathcal{S}_\mathsf{V}$ invokes the simulator of Π_{OT} for corrupt receiver to simulate the first OT message.

GC Construction and OT Second Message Phase: $\mathcal{S}_\mathsf{V}$ uses the OT simulator to extract V's inputs to the jth instance of Π_{OT}, namely (k_j^0, k_j^1).

GC Evaluation, Verification and Output Disclosure Phase: On receiving the garbled circuit $\mathbf{C}$ and T from V, $\mathcal{S}_\mathsf{V}$ runs $\mathsf{Ve}_2(C, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]})$. It aborts if the output of Ve_2 is 0. Else, it sends k^1 to V where $k^1 \leftarrow \mathsf{Ve}_1(\mathbf{C}, e, 1)$.

Output Phase: It does nothing in this step.

Fig. 5: Simulator $\mathcal{S}_\mathsf{V}$

Security against a Corrupt Verifier $\mathsf{V}^\star$. We now argue that $\text{IDEAL}_{\mathcal{F}_{\mathsf{ZK}}^R,\mathcal{S}_\mathsf{V},\mathcal{Z}} \overset{c}{\approx} \text{REAL}_{\Pi_{\mathsf{ZK3}},\mathcal{A},\mathcal{Z}}$ when $\mathcal{A}$ corrupts V. The above two views of V^* are shown to be indistinguishable via a series of intermediate hybrids.

- HYB_0: Same as $\text{REAL}_{\Pi_{\mathsf{ZK3}},\mathcal{A},\mathcal{Z}}$.
- HYB_1: Same as HYB_0, except that **OT First Message phase** is emulated by invoking the simulator of Π_{OT} for corrupt receiver.
- HYB_2: Same as HYB_1, except that k^1 is computed in the following way instead of running $\mathsf{Ev}(\mathbf{C}, \mathbf{X})$. The simulator of Π_{OT} for corrupt receiver is used to extract (k_j^0, k_j^1) for $j \in [n]$. Then $\mathsf{Ve}_2(C, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]})$ is run. If the output is 0, the prover aborts. Otherwise $\mathsf{Ve}_1(\mathbf{C}, e, 1)$ is run to extract k^1 and the prover runs the rest of the protocol using k^1.
- HYB_3: Same as HYB_2, except that the following check for abort in **GC Evaluation, Verification and Output Disclosure Phase** is removed: On computing $r_1^\mathsf{S}||\cdots||r_n^\mathsf{S} = r^\mathsf{S} = T \oplus H(k^1)$, the prover aborts if any call to the extractor Ext of the sender's input to OT returns $\perp$.

Clearly, $\text{HYB}_3 = \text{IDEAL}_{\mathcal{F}_{\mathsf{ZK}}^{R},\mathcal{S}_{\mathsf{V}},\mathcal{Z}}$. Our proof will conclude, as we show that every two consecutive hybrids are computationally indistinguishable.

$\text{HYB}_0 \stackrel{c}{\approx} \text{HYB}_1$: The difference between these hybrids lies in the way OT first message is generated. In HYB_0, the message is generated by a real receiver that possesses the choice bits x, whereas in HYB_1, the simulator for Π_{OT} for the corrupt receiver generates the message. The indistinguishability follows via reduction to the sender security of n instances of OT.

$\text{HYB}_1 \stackrel{c}{\approx} \text{HYB}_2$: The difference between these hybrids lies in the way k^1 is computed. In HYB_1, k^1 is computed as a real prover does. On the other hand, k^1 is extracted using Ve_1 and the encoding information extracted from the OTs in HYB_2. By the verifiability property (Definition A.5) of the garbling scheme, the view of $\mathsf{V}^\star$ in HYB_2 and HYB_1 are indistinguishable.

$\text{HYB}_2 \stackrel{c}{\approx} \text{HYB}_3$: The difference between these hybrids lies in the conditions checked by P for abort in **GC Evaluation, Verification and Output Disclosure Phase**. In the former, the protocol is aborted when one of the invocations to Ext returns messages different from corresponding input labels which does not happen in the latter as the check is removed. By the sender extractability of the OT protocol (Definition 4.1), the hybrids are indistinguishable except with negligible probability.

□

4.3 Making Π_{ZK3} Adaptively Secure

The challenge in achieving adaptive security for Π_{ZK3} is essentially the same as ZKGC; once the GC output key Z has been retreived, all of P's steps are independent of the witness.

Simulation for P. Consider the worst case scenario of post-execution corruption. The simulator runs $\left(m^{\mathsf{R}}, r_0^{\mathsf{R}}, r_1^{\mathsf{R}}\right) \leftarrow \mathcal{S}^{\mathsf{RE}}(\mathsf{crs}, t)$ to generate the first message of P, and obtains the GC output key Z either by extracting the encoding information from V's response (if V is corrupt) or using the key it picked itself when simulating V. The rest of the simulation is straightforward, as the code of honest P can be run from this point. In case the adversary chooses to corrupt P, the simulator hands over the randomness $r_{x_i}^{\mathsf{R}}$ for each OT instance encoding witness bit x_i.

Simulation for V. As V has no input, the simulator proceeds by running the code of the honest verifier, with the only difference being that it accepts a proof by checking whether P has input a valid witness in the OT. A malicious P can distinguish between the real protocol and the simulation only by forging Z, for which there is no advantage afforded by adaptive corruptions; a dishonest P who is successful in this setting can be used to break authenticity of the garbling scheme just as in the static case.

5 Zero Knowledge in Two Rounds

As discussed in Section 4, it seems unlikely that we can do better than three rounds to obtain a zero-knowledge from only one garbled circuit. Therefore, we explore whether we can save on the number of rounds when constructing ZK protocols by allowing multiple garbled circuits. In this section, we adopt a 'cut-and-choose' approach in order to construct a GC-based ZK protocol that requires *only* two rounds.

Our protocol is similar in spirit to the protocol of [HV16], who extend the technique of "MPC-in-the-head" [IKOS07]. The "MPC-in-the-head" is a technique introduced by Ishai et al. that allows a generic transformation of an MPC protocol into a zero-knowledge proof. In [HV16], the authors extend this idea, and give a generic transformation from a secure two-party computation protocol to a ZK proof.

The protocol is essentially a special case of general cut-and-choose. Since the verifier has no input, we do not have to handle selective failure where the evaluator's abort could leak a bit of his input, or ensure input consistency of the garbler, again, since the circuit is evaluated on an input entirely known to the garbler. While in [HV16], the protocol is seen as "2PC-in-the-head", we cast our protocol as cut-and-choose, and apply a standard transformation based on OT. Loosely speaking, choosing to reveal P_1's view in "2PC-in-the-head" in [HV16] is equivalent to choosing a circuit to be a check circuit in our protocol; and choosing to reveal P_2's view corresponds to a circuit being an evaluation circuit. Taking this view, we get a zero-knowledge argument whereas the "2PC-in-the-head" of [HV16] gives a zero-knowledge proof. We note that we do not need to enforce output recovery when two evaluated circuits result in different outputs. The output recovery mechanism that is used in general 2PC protocols [Lin13,LR14,LR15,AMPR14,MR17] relies on authenticity property of the underlying garbling scheme. Our protocol can be compiled into a NIZK using standard techniques and transformations.

Next, we note that we can upgrade our argument to a proof following the idea of [HV16]; we augment our two-round argument with statistically binding commitments to the input GC keys from P. The inputs of P to the OT consist of the openings of all commitments (for a check circuit) as one message, and only the committed keys required to evaluate the GC on the garbled witness as the other message. Notably, the efficient ZK protocols such as those from garbled circuits [JKO13] (including our 3 round construction presented in the previous section), ZKBoo [GMO16], SNARKs and SNARGs are arguments. Our transformation requires public key operations proportional to the witness size alone whereas the best way we can think of for transforming ZKBoo to a proof involves public key operations proportional to the circuit size. For instance, running a 3-out-of-2 OT where the prover feeds three views that it creates 'in the head' as the input of the OT sender and the verifier chooses two indices picked uniformly at random indicating the two views to be opened for verification.

Once more, we consider the scenario where a prover P would like to prove to a verifier V that she knows a witness x for instance z such that $C(x) = 1$, where C is the circuit implementing the relation $R(z, x)$.

5.1 Our Construction

Informally, P garbles C to produce μ independent garbled circuits, and sends them to V, where μ is a statistical security parameter. Meanwhile, V samples a challenge string $c \xleftarrow{R} \{0,1\}^\mu$. The positions at which bit string c is 0 will indicate which circuits V would like to verify (check circuits), whereas the positions at which c is 1 indicate which circuits V would like to evaluate (evaluation circuits). If all the check circuits are valid, and all the evaluation circuits decode to the correct output, V believes that P indeed has a witness x for the instance z. P would have to correctly guess V's entire challenge string in order to cheat and avoid detection.

Intuitively, P constructs μ independent garbled circuits of C, and for each instance acts as a sender in the OT protocol with messages corresponding to verification and evaluation information, respectively of the garbled circuit $\mathbf{C}$, while sending the garbled circuit and decoding information directly to V (with the final message of the OT). V acts as the receiver in the OT protocol with choice bit c_i in the i^{th} OT instance. She receives the first message to check or the second message to evaluate a given circuit, as per her challenge. When instantiated with the UC-secure OT in the framework of [PVW08], our protocol requires only 2 rounds. Our 2-round ZK protocol Π_{ZK2} is described in Fig. 6, and proven UC-secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model in Appendix D.

The zero knowledge protocol Π_{ZK2} is not a zero knowledge proof. It is only an argument. We may obtain a proof using the idea of [HV16], resulting in a 2-round zero-knowledge proof. We outline the approach below for completeness.

5.2 Our construction for ZK proof

The zero knowledge protocol Π_{ZK2} is not a zero knowledge proof. It is only an argument. We may obtain a proof using the idea of [HV16], resulting in a 2-round zero-knowledge proof. For a legitimately constructed garbled circuit $\mathbf{C}$ implementing an unsatisfiable circuit (implying there is no witness for the statement), an unbounded $\mathsf{P}^\star$ can find a set of keys, completely unrelated to the legitimate encoding information e, (say, by breaking the security of the underlying cryptographic primitive used in the garbled circuit) which evaluates $\mathbf{C}$ to the legitimate key corresponding to one. For instance, by breaking the collision-resistance of the hash function used to garble the gates. With such a circuit, the verification will always pass when legitimate encoding information is passed on. On the other hand, the other set of keys will allow to evaluate to 1 despite the fact that C is unsatisfiable. $\mathsf{P}^\star$ can thus convince V of a false statement. To prevent P from cheating we ensure that the wire labels that it provides for evaluation correspond to the valid encoding information e. This is done by asking P to commit

Π_{ZK2}

- **Oracles and Cryptographic Primitives:** A *correct, private, verifiable* (according to Definition A.4) garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{Ve})$. The ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$.
- **Common Inputs of P and V:** A security parameter λ, soundness parameter μ, relation R realized by circuit C, statement z.
- **Input of P:** A witness x of size $n = \mathsf{poly}(\lambda)$ such that $R(z, x) = 1$.
- **Input of V:** Nothing.

OT First Message Phase: For all $i \in [\mu]$, V samples challenge bit $c_i \overset{R}{\leftarrow} \{0,1\}$ and sends $(\mathtt{rec}, \mathrm{sid}, c_i)$ to $\mathcal{F}_{\mathsf{OT}}$.

OT Second Message and Circuit Communication Phase: For all $i \in [\mu]$, P
- constructs an independent garbling of C; $(\mathbf{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\lambda, C)$
- encodes the witness as $\mathbf{X}_i = \mathsf{En}(x, e_i)$
- sends $(\mathtt{sen}, \mathrm{sid}, e_i, \mathbf{X}_i)$ to $\mathcal{F}_{\mathsf{OT}}$ and $(\mathbf{C}_i, d_i)$ to V.

Circuit Checking, Evaluation and Output Phase: This is a local computation phase run by V. For all $i \in [\mu]$, V does the following:
- If $c_i = 0$, then it receives $(\mathtt{sent}, \mathrm{sid}, e_i)$ from $\mathcal{F}_{\mathsf{OT}}$. If $\mathsf{Ve}(C, \mathbf{C}_i, e_i) = 0$, then it outputs reject and halt.
- Else if $c_i = 1$, then it receives $(\mathtt{sent}, \mathrm{sid}, \mathbf{X}_i)$ from $\mathcal{F}_{\mathsf{OT}}$. If $\mathsf{De}(\mathsf{Ev}(\mathbf{C}_i, \mathbf{X}_i), d_i) \neq 1$ then it outputs $\mathtt{reject}$ and halt.
- If it has not halted, it outputs $\mathtt{accept}$.

Fig. 6: 2-round Zero-Knowledge protocol.

to the encoding information in a randomly permuted order. Formally, for circuit i and input wire j, P must prepare and send the following commitments where e_{ij} denotes the encoding information corresponding to jth input wire of the ith circuit:

$$(\mathcal{B}_{ij}^0, \mathcal{B}_{ij}^1) = (\mathsf{Com}(\mathsf{En}(b_{ij}, e_{ij})), \mathsf{Com}(\mathsf{En}(1 - b_{ij}, e_{ij}))), \text{ for } b_{ij} \overset{R}{\leftarrow} \{0,1\}$$

The commitment Com is statistically binding and computationally hiding commitment scheme ensuring the binding property against an unbounded powerful $\mathsf{P}^\star$. An ElGamal based commitment scheme suffices for our requirement. V checks if the commitments $(\mathcal{B}_{ij}^0, \mathcal{B}_{ij}^1)$ opens to legitimate encoding information if the ith circuit is a check circuit. On the other hand, if the ith circuit is an evaluation circuit, then it verifies that every received input wire label is consistent with one of the given commitments. The commitments used as above makes sure that V evaluates the evaluation circuits on the legitimate wire labels consistent with e. The cut-and-choose guarantees that correct circuits are used for evaluation. Our protocol is presented in Figure 7 and a schematic diagram is given in Figure 17. The security proof for the scheme is an easy extension of the proof given in the previous section for the ZK argument.

Π_{ZKP2}

- **Oracles and Cryptographic Primitives:** A *correct, private, verifiable* (according to Definition A.4) garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{Ve})$. The ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$. Statistically binding and computationally hiding commitment scheme Com.
- **Common Inputs of P and V:** A security parameter λ, soundness parameter μ, relation R realized by circuit C, statement z.
- **Input of P:** A witness x of size $n = \mathsf{poly}(\lambda)$ such that $R(z, x) = 1$.
- **Input of V:** Nothing.

OT First Message Phase: For all $i \in [\mu]$, V samples challenge bit $c_i \xleftarrow{R} \{0,1\}$ and sends $(\mathtt{rec}, \mathrm{sid}, c_i)$ to $\mathcal{F}_{\mathsf{OT}}$.

OT Second Message and Circuit Communication Phase: For all $i \in [\mu]$, P
- constructs an independent garbling of C; $(\mathbf{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\lambda, C)$
- encodes the witness as $\mathbf{X}_i = \mathsf{En}(x, e_i)$
- for $j \in [n]$, samples $b_{ij} \xleftarrow{R} \{0,1\}$, generates commitments as $(\mathcal{B}^0_{ij}, \mathcal{B}^1_{ij}) = (\mathsf{Com}(\mathsf{En}(b_{ij}, e_{ij})), \mathsf{Com}(\mathsf{En}(1 - b_{ij}, e_{ij})))$ on the encoded inputs for input wire j (e_{ij} denotes the encoding information corresponding jth input wire of the ith circuit)
- sends $(\mathtt{sen}, \mathrm{sid}, (e_i, r_i), (\mathbf{X}_i, s_i))$ to $\mathcal{F}_{\mathsf{OT}}$ and $(\mathbf{C}_i, d_i)$ and $(\mathcal{B}^0_{ij}, \mathcal{B}^1_{ij})$ for $j \in [n]$ to V (where r_i is the opening information for all the commitments used for ith circuit and s_i is the opening information for only the commitment used for the keys encoding x).

Circuit Checking, Evaluation and Output Phase: This is a local computation phase run by V. For all $i \in [\mu]$, V does the following:
- If $c_i = 0$, then it receives $(\mathtt{sent}, \mathrm{sid}, (e_i, r_i))$ from $\mathcal{F}_{\mathsf{OT}}$. If $\forall j \in [n]$, (e_i, r_i) is not consistent with $(\mathcal{B}^0_{ij}, \mathcal{B}^1_{ij})$ or $\mathsf{Ve}(C, \mathbf{C}_i, e_i) = 0$ then V outputs reject and halt.
- Else if $c_i = 1$, then it receives $(\mathtt{sent}, \mathrm{sid}, (\mathbf{X}_i, s_i))$ from $\mathcal{F}_{\mathsf{OT}}$. If $\forall j \in [n]$, $(\mathbf{X}_i, s_i)$ is not consistent with $\mathcal{B}^{b_{ij}}_{ij}$ for some $b_{ij} \in \{0,1\}$ or $\mathsf{De}(\mathsf{Ev}(\mathbf{C}_i, \mathbf{X}_i), d_i) \neq 1$ then V outputs $\mathtt{reject}$ and halt.
- If it has not halted, it outputs $\mathtt{accept}$.

Fig. 7: 2-round protocol for obtaining Zero-Knowledge Proof.

6 On Authenticity-Free Garbling

As we have seen earlier, garbling schemes that achieve privacy alone are well-motivated. Along the lines of [FNO15] one might ask the question, *"can we leverage the lack of an authenticity requirement in order to construct more efficient garbling schemes?"*

Unfortunately, in this section, we answer the above question in the negative, for most 'standard' garbling schemes. Note that more efficient garbled circuit constructions by giving up on authenticity is precluded by the result of [AIK10]. The result of [AIK10] shows how to achieve authenticity from a garbling scheme that satisfies only privacy. While this involves encoding an augmented circuit, we show a similar result while encoding the *same* underlying circuit. Specifically, we show that if a garbling scheme achieving privacy satisfies a notion of *compos-*

ability, then the scheme is necessarily authentic. Intuitively, any garbling scheme that does not treat output gates differently from intermediate/input gates will be composable. This definition covers state-of-the-art constructions such as those of [ZRE15,GLNP15], and the basic Yao garbling scheme itself [LP09]. We show that an adversary who is able to forge a garbled output for a GC produced by a composable garbling scheme, can be used to break privacy of a slightly larger circuit garbled by the same scheme. Intuitively, forging the output key of the sub-circuit enables multiple evaluations on the larger circuit. As a simulated garbled circuit is constructed to simulate only one 'path' of garbled evaluation, it will most likely not permit multiple correct evaluations. We begin with the formal definition of composable garbling in the section below. In Section 6.3, we present how an AND gate can be garbled with just one ciphertext non-composably.

6.1 Composable Garbling

We restrict our model to projective garbling schemes which work by associating keys k_i^0, k_i^1 corresponding to semantic values 0 and 1 respectively for each input/output wire i in the circuit being garbled. Note that referring to the key corresponding to semantic 0 as k_i^0 is done for notational convenience; our proof is unaffected by point-and-permute style optimizations. We further focus on the garbling schemes where the decoding algorithm De satisfies the following property: if De does not output $\perp$ for a given garbled output $\mathbf{Y}$, then the garbled output keys that comprise $\mathbf{Y}$ are necessarily drawn from the pre-defined output wire keys. Namely, for $\mathbf{Y} = (\mathbf{Y}_i)_{i\in[n+q+1,n+q+m]}$:

$$\mathsf{De}\,(\mathbf{Y}, d) \neq \perp \implies \forall i \in [n+q+1, n+q+m],\ \mathbf{Y}_i \in \{k_i^0, k_i^1\}$$

where n, q, m denote the number of input wires, gates and output wires respectively in circuit C. Placing the above requirement allows us to ignore garbling schemes that do not achieve authenticity only because the De routine never outputs $\perp$ [5].

Informally, a composable garbling scheme allows the output keys of a previously garbled gate to be the input keys to an instance of garbling another gate in the circuit. A Gb routine that directly uses the output keys of a gate as the input keys to a child gate, and does not distinguish between output and non-output gates, will make the garbling scheme composable. A bit formally, a scheme Garble is said to be composable if it can be used to define a new garbling scheme $\mathsf{Garble}^\star$ that can compose garbling of a circuit C and a gate C' according to Gb of Garble to produce a GC for a larger composed circuit $C^\star$ defined as follows. Circuit $C^\star : \{0,1\}^n \mapsto \{0,1\}^m$ is interpreted as the composition of the sub-circuit $C : \{0,1\}^{n-1} \mapsto \{0,1\}^m$ and C'. One of the output wires of C acts as the left input wire of gate C'. The last input wire of $C^\star$ serves as the right input wire of C'. C' provides an output wire in circuit $C^\star$.

[5] Such a pathological scheme is used to show separation between authenticity and the other security notions in [BHR12b]

Our way of defining composability is sufficient for the purpose of our proof, while capturing most practical garbling schemes [ZRE15,GLNP15]. We note that the template of Gb of the scheme $\mathsf{Garble}^\star$ can be extended to more accurately reflect garbling for general composed circuits by removing the constraints on the input wires of C'. Namely, that the right wire of C' is an input wire, whereas the left wire is an intermediate wire. While not required for our proof, we provide such a template capturing gate-by-gate garbling schemes in Section 6.4 for completeness. The tradeoff for a more precise template is a loss of generality in the garbling techniques captured; as an example, the work of [MPs16] abandons the gate-by-gate approach to garbling. The composability requirement as stated formally in Fig. 8 for our proof is meaningful for any projective garbling scheme, which we believe will be relevant beyond current garbling techniques. We now formalize the notion of composable garbling scheme.

Definition 6.1 *A garbling scheme* $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is* composable *if there exists a garbling routine* $\mathsf{Gb}^\star$ *as per Fig. 8 such that the composed garbling scheme* $\mathsf{Garble}^\star = (\mathsf{Gb}^\star, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is* correct *and* private.

Given a garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ of a garbling scheme Garble, we construct a garbling algorithm $\mathsf{Gb}^\star$ for a given circuit $C^\star : \{0,1\}^n \mapsto \{0,1\}^m$, which is subject to certain restrictions as given below. Circuit $C^\star$ is interpreted as the composition of a sub-circuit $C : \{0,1\}^{n-1} \mapsto \{0,1\}^m$ and a single 2 fan-in gate C'. The gate C' provides an output wire in circuit $C^\star$. The left and right incoming wires to C' are indexed L and R respectively. Clearly L is an output wire of C, and R is an input wire of $C^\star$.

$$\mathsf{Gb}^\star\left(1^\lambda, C^\star\right)$$

1. Parse C and C' from $C^\star$, where C' is the last gate in $C^\star$.
2. Use Gb to garble C, i.e. $(\mathbf{C}, e, d) \leftarrow \mathsf{Gb}\left(1^\lambda, C\right)$
3. Extract keys k_L^0, k_L^1 using $(C, \mathbf{C}, e)^a$.
4. Choose fresh keys k_R^0, k_R^1. For garbling schemes such as FreeXOR which require a certain key structure, choose fresh input keys appropriately. Otherwise, two independent random λ-bit strings will suffice.
5. Compute $(\mathbf{C}', e', d') = \mathsf{Gb}\left(1^\lambda, C'\right)$ such that $e' = \left(\left(k_L^0, k_L^1\right), \left(k_R^0, k_R^1\right)\right)$
6. Set $\mathbf{C}^\star = \mathbf{C}||\mathbf{C}'$, $e^\star = e||e'[2]$ and $d^\star = d[1]||d[2]|| \cdots ||d[m-1]||d'$ where e and e' are perceived as arrays with ith entry containing the encoding information for the ith input wire. Similar notation is used for d and d'.
7. **return** $\mathbf{C}^\star, e^\star, d^\star$

[a] This can be done by saving the required keys from Step 2

Fig. 8: Specification of a composing Gb routine

6.2 Relating Composability and Privacy to Authenticity

We now show that a garbling scheme Garble whose composed garbling scheme $\mathsf{Garble}^\star$ is private and correct as per Def. 6.1 must necessarily be authentic.

Theorem 6.2 *If a* private *and* correct *garbling scheme* $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ *is* composable *as per Definition 6.1, then it is also* authentic*.*

Proof. Given black-box access to an adversary $\mathcal{A}_{\mathsf{aut}}$ who is able to forge a garbled output for some (C, x) garbled using Garble, we construct an adversary $\mathcal{A}_{\mathsf{prv}}$ who can distinguish between a legitimate and simulated garbling of some larger circuit $(C^\star, x||x')$ as per composed garbling scheme $\mathsf{Garble}^\star$ (as per Def. 6.1). The idea is to create $C^\star$ from C so that the output wire of C is an intermediate wire of a circuit $C^\star$. $\mathcal{A}_{\mathsf{prv}}$ then uses $\mathcal{A}_{\mathsf{aut}}$ to forge the missing output key of the output of C. Now, possession of both the keys of the output wire of C allows $\mathcal{A}_{\mathsf{prv}}$ to perform multiple evaluations of output gate C' in a garbling of $C^\star$. This in turn allows to verify if the gate has been garbled legitimately or via a simulator.

Assuming black box access to an adversary $\mathcal{A}_{\mathsf{aut}}$ who can forge a garbled output for some circuit $C : \{0,1\}^{n-1} \mapsto \{0,1\}$ and input $x \in \{0,1\}^{n-1}$ as per Garble, we construct adversary $\mathcal{A}_{\mathsf{prv}}$ who can distinguish whether a given $(\mathbf{C}, \mathbf{X}, d)$ corresponding to a larger circuit $C^\star : \{0,1\}^n \mapsto \{0,1\}$ as per $\mathsf{Garble}^\star$ is produced legitimately, or simulated.

$$\mathcal{A}_{\mathsf{prv}}\left(1^\lambda\right)$$

1. **receive** C, x from $\mathcal{A}_{\mathsf{aut}}$. Let y be the output wire label of C.
2. Construct $C^\star$ as follows:
   - **if** $C(x) = 0$ **then** $C^\star = C||(y \wedge x')$
   - **else** $C^\star = C||(\neg y \wedge x')$
3. Sample bit $x' \xleftarrow{R} \{0,1\}$
4. **send** $C^\star, x||x'$ as the function and input to the challenger of the privacy experiment, and **receive** $\mathbf{C}^\star, \mathbf{X}^\star, d^\star$ as a response.
5. From the above response, parse $\mathbf{C}||\mathbf{C}' = \mathbf{C}^\star$, where $\mathbf{C} \in \mathsf{Gb}(1^\lambda, C)$ and $\mathbf{C}' \in \mathsf{Gb}(1^\lambda, \wedge)$, as well as $\mathbf{X}||\mathbf{X}' = \mathbf{X}^\star$ where $\mathbf{X} \in \mathsf{En}(x, \cdot)$ and $\mathbf{X}' \in \mathsf{En}(x', \cdot)$
6. Compute $\mathbf{Y} = \mathsf{Ev}\left(\mathbf{C}, \mathbf{X}\right)$
7. **send** $(\mathbf{C}, \mathbf{X})$ to $\mathcal{A}_{\mathsf{aut}}$ and **receive** $\mathbf{Y}'$ as a candidate forged output.
8. Compute $\mathbf{Z} = \mathsf{Ev}\left(\mathbf{C}', \mathbf{Y}||\mathbf{X}'\right)$ and $\mathbf{Z}' = \mathsf{Ev}\left(\mathbf{C}', \mathbf{Y}'||\mathbf{X}'\right)$
9. Check the consistency of the final garbled gate to determine whether it was garbled legitimately, as follows:
   - **if** $\mathsf{De}\left(\mathbf{Z}', d\right) = \perp$ **then output** $guess \xleftarrow{R} \{\text{Real}, \text{Ideal}\}$
   - **else if** ($\mathbf{Z}' = \mathbf{Z}$ and $x' = 0$) or ($\mathbf{Z}' \neq \mathbf{Z}$ and $x' = 1$) **then output** $guess =$ Real
   - **else output** $guess =$ Ideal

Fig. 9: Reduction of authenticity of Garble to privacy of Garble'

Concretely, $(C^\star, x||x')$ is constructed from (C, x) as follows. If $C(x) = 0$, then $C^\star$ is constructed as C appended with an AND gate that takes the output of C

as the left input and x' as the right input. Otherwise, negation of the output of C is given as the left input to the AND gate. x' is chosen uniformly at random. The above construction of $C^\star$ ensures that the output of $C^\star$ is always 0. This completely hides from the garbling simulator $\mathcal{S}$ the value that x' takes, thereby providing no clues for $\mathcal{S}$ to use the keys of the wire in the right way. As we analyse below, the best way for the simulator to escape being told apart from a legitimate garbling is to guess x'. x' being randomly picked ensures a probability of $\frac{1}{2}$ for the simulated garbled circuit being concluded as a legitimate garbled circuit.

This reduction of the authenticity of garbling scheme Garble to the privacy of composed garbling scheme $\mathsf{Garble}^\star$ is formally described in Fig. 9, with a formal analysis below. For simplicity, we assume that C outputs only 1 bit; we will address the general case later.

Let the advantage of $\mathcal{A}_{\mathsf{aut}}$ in correctly forging a valid missing garbled output for a GC and input produced by Garble be $h(\lambda)$ (see Def. A.3). Index the Real world $b = 0$, and Ideal world $b = 1$. We analyze the following cases in order to determine how much of this advantage is translated in distinguishing a simulated $(\mathbf{C}, \mathbf{X}, d)$ from one legitimately produced by $\mathsf{Garble}^\star$:

Real world: $(\mathbf{C}^\star, \mathbf{X}^\star, d^\star) \in \{\mathbf{Real}_{\mathsf{Garble}^\star}(C^\star, x||x')\}$.

a. $\mathcal{A}_{\mathsf{aut}}$ *is successful in forging an output.* This case occurs with probability $h(\lambda)$. Then, $\mathcal{A}_{\mathsf{prv}}$ guesses Real and is correct with probability 1.
b. $\mathcal{A}_{\mathsf{aut}}$ *is unsuccessful in forging an output.* This case occurs with probability $1 - h(\lambda)$. Then, either decoding or evaluation of the final gate will fail; $\mathcal{A}_{\mathsf{prv}}$ submits a random guess and is correct with probability $\frac{1}{2}$.

In this case, the adversary $\mathcal{A}_{\mathsf{prv}}$ correctly guesses that she is in the Real world with the following probability:

$$\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^\lambda\right) = 0 \mid b = 0\right] = h(\lambda) \cdot 1 + (1 - h(\lambda)) \cdot \frac{1}{2} = \frac{1}{2} \cdot (h(\lambda) + 1) \qquad (1)$$

Ideal world: $(\mathbf{C}^\star, \mathbf{X}^\star, d^\star) \in \{\mathbf{Ideal}_{\mathcal{S}}(C^\star, 0)\}$.

a. $\mathcal{A}_{\mathsf{aut}}$ *is successful in forging an output.* This case occurs with some probability $h'(\lambda)$. We will give the benefit of the doubt to $\mathcal{S}$ and assume that all keys and ciphertexts in $\mathbf{C}^\star$ are consistent, and that the decoding information is constructed correctly (if not, $\mathcal{A}_{\mathsf{prv}}$ will terminate with a random guess, winning with probability $\frac{1}{2}$). Now if $\mathcal{S}$ can guess x' and correctly pass the corresponding key, the garbled gate for the final gate will pass the consistency check and $\mathcal{A}_{\mathsf{prv}}$ will output Real. Otherwise, $\mathcal{A}_{\mathsf{prv}}$ will output Ideal. The probability of guessing x' is at most $\frac{1}{2}$ since $\mathcal{S}$ has no information about x' whatsoever (as the function output is always 0) and hence can at best guess x' at random. Therefore, $\mathcal{A}_{\mathsf{prv}}$ outputs Ideal with probability $\frac{1}{2}$.
b. $\mathcal{A}_{\mathsf{aut}}$ *is unsuccessful in forging an output.* This case occurs with probability $1 - h'(\lambda)$. Then, either decoding or evaluation of the final gate will fail; $\mathcal{A}_{\mathsf{prv}}$ submits a random guess and is correct with probability $\frac{1}{2}$.

The adversary $\mathcal{A}_{\mathsf{prv}}$ therefore has the following probability in guessing correctly that she is in the ideal world:

$$\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\lambda}\right) = 1 \mid b = 1\right] = h'\left(\lambda\right) \cdot \frac{1}{2} + \left(1 - h'\left(\lambda\right)\right) \cdot \frac{1}{2} = \frac{1}{2} \tag{2}$$

We can compute the advantage of $\mathcal{A}_{\mathsf{prv}}$ in distinguishing the output distributions $\{\textsc{Real}_{\mathsf{Garble}^{\star}}\left(C, x||x'\right)\}$ and $\{\textsc{Ideal}_{\mathcal{S}}\left(C, 0\right)\}$, plugging in the probabilities from (1) and (2) as follows:

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\lambda}\right) = 0 \mid b = 0\right] - \mathbf{Pr}\left[\mathcal{A}_{\mathsf{prv}}\left(1^{\lambda}\right) = 0 \mid b = 1\right] &= \frac{1}{2} \cdot \left(h\left(\lambda\right) + 1\right) - \frac{1}{2} \\ &= \frac{1}{2} \cdot h\left(\lambda\right)\end{aligned} \tag{3}$$

Given that Garble is composable, $\mathsf{Garble}^{\star}$ is private by definition. As per Def. A.2, the advantage of $\mathcal{A}_{\mathsf{prv}}$ computed in (3) must be negligible, implying that $h\left(\lambda\right)$ must be negligible for all $\mathcal{A}_{\mathsf{aut}}\left(1^{\lambda}\right)$. There can not exist a PPT adversary $\mathcal{A}_{\mathsf{aut}}$ who can succesfully forge a garbled output for any $\mathbf{C}, \mathbf{X}$ produced by a composable garbling scheme Garble with non-negligible advantage as per Definition A.3. Therefore, given that a garbling scheme Garble is composable, it is necessarily authentic, and this proves Theorem 6.2. □

What if C has multiple wires? To handle the case where C of the tuple (C, x) has m output wires denoted as $y_1, \ldots, y_m$, we define $C^{\star} : \{0,1\}^{n+m} \mapsto \{0,1\}^{m}$ as C appended with m AND gates where ith AND gate takes either y_i or $\neg y_i$ (adjusting as per Step 2 of Fig. 9) as the left input and x'_i as the right input wire. The strategy of $\mathcal{A}_{\mathsf{prv}}$ hence follows with at least the same advantage, as if $\mathcal{A}_{\mathsf{aut}}$ forges output keys on m' wires, $\mathcal{S}$ is successful only when it guesses every x'_i input to the corresponding wires, which it can do with probability no greater than $2^{-m'}$.

6.3 Feasibility of Authenticity-Free Garbling

Garbling gate by gate in topological order where the output keys of one garbled gate are used as the input keys to its children, coupled with circuit output key distributions being (nearly) identical to the input key distribution of intermediate gates, is the dominant paradigm underlying most state-of-the-art garbling schemes for Boolean circuits [ZRE15,GLNP15,KMR14,KS08,PSSW09]. This means that the current methods of garbling Boolean circuits privately is inherently composable, therefore making authenticity impossible to avoid.

However, we note that it is possible to have efficient authenticity-free garbling that is non-composable, assuming access to a PRF F. Consider any projective topological gate-by-gate garbling scheme Garble with the following modifications to construct $\mathsf{Garble}^{\star}$:

– Gb : Garble all gates until the output layer topologically as per Garble; the keys on wire i are k_i^0, k_i^1 corresponding to semantic values 0 and 1, and the ciphertext for wire i is stored in $T[i]$. Let $i \in [m]$ index the output gates, while L_i and R_i index the left and right incoming wires of i respectively. For all $i \in [m]$ such that i is an AND gate, set ciphertext $T[i] := F_{k^1_{L_i}}(i) \oplus F_{k^1_{R_i}}(i)$, zero-key $k_i^0 := 0^\lambda$ and one-key $k_i^1 := 1^\lambda$.
– Ev : Evaluate all gates until the output layer topologically as per Garble; the key obtained on wire w is k_w. Let $i \in [m], L_i, R_i$ be defined as earlier. Evaluating output AND gate $i \in [m]$ proceeds as follows: compute $C := F_{k_{L_i}}(i) \oplus F_{k_{R_i}}(i)$. If $C = T[i]$ then set $k_i = 1^\lambda$, otherwise set $k_i = 0^\lambda$.

Security. The output keys for any output-layer AND gate are predictably always 0^λ and 1^λ, making $\mathsf{Garble}^\star$ clearly non-authentic. These output keys can not be reused as input keys to another gate for the same reason, making $\mathsf{Garble}^\star$ non-composable. However if Garble is private and correct, then so is $\mathsf{Garble}^\star$; leaking the semantic values of the output keys to the evaluator does not compromise privacy. The privacy property of a garbling scheme requires that a PPT $\mathcal{A}$ can not (with non-negligible advantage) distinguish between an honestly constructed $(\mathbf{C}, \mathbf{X}, d)$ and such values constructed by a simulator that has access to only the clear output $C(x)$ (and not x). $\mathcal{A}$ is allowed to see the decoded output anyway, therefore the distribution of the routines of $\mathsf{Garble}^\star$ can be simulated for privacy if those of Garble can be. The Garble1 scheme of [BHR12b] achieves privacy despite leaking the semantic values of the output wires.

Performance. Most practical garbling schemes are shown to satisfy a definition of 'linearity' by Zahur et al. [ZRE15]. They go on to show that a linear garbling scheme achieving privacy requires at least two ciphertexts to garble an AND gate. However, $\mathsf{Garble}^\star$ garbles every output AND gate with just one ciphertext, implying that for any linear Garble, a corresponding $\mathsf{Garble}^\star$ as defined above will necessarily produce *one less ciphertext* per output AND gate. Our result indicates that any approach to authenticity-free garbling must be inherently non-composable as per our definition. However, the result of [AIK10] suggests that even non-composable garbling schemes might not gain much by giving up on authenticity.

Lower bound of [ZRE15]. The garbling scheme $\mathsf{Garble}^\star$ when used to garble a single AND gate in isolation produces only one ciphertext, which may seem to contradict the 2-ciphertext lower bound for private garbling proven in [ZRE15]. However the Ev routine of of $\mathsf{Garble}^\star$ makes use of a comparison operation, which disqualifies it from being a linear garbling scheme. Therefore, instead of a counterexample, we have a circumvention of the 2-ciphertext lower bound.

6.4 Composable Gate-by-gate Garbling

Our template for gate-by-gate composable garbling is detailed in Fig. 10.

For garbling scheme Garble, we provide a template for a garbling routine $\mathsf{Gb}^\star$ which uses the garbling routine $\mathsf{Gb} \in \mathsf{Garble}$ to garble a given circuit $C^\star$. A circuit $C^\star$ is interpreted as the composition of a sub-circuit C and a single 2 fan-in gate C'. The gate C' provides an output wire in circuit C. The left and right incoming wires to C' are indexed L and R respectively. Note that L and R could each be any of the following: an input wire to C, an internal wire in C, or an input wire of $C^\star$ that does not touch C at all.

$$\mathsf{Gb}^\star\left(1^\lambda, C^\star\right)$$

1. Parse C and C' from $C^\star$, where C' was indexed as the last gate in C
2. Use Garble to garble C. $\mathbf{C}, e, d \leftarrow \mathsf{Gb}\left(1^\lambda, C\right)$
3. For $w \in \{L, R\}$ such that w is a wire touching C, extract keys k_w^0, k_w^1 using $(C, \mathbf{C}, e)^a$.
4. For $w \in \{L, R\}$ such that w is a wire not touching C, choose fresh keys k_w^0, k_w^1. For garbling schemes such as FreeXOR which require a certain key structure, choose fresh input keys appropriately. Otherwise, two independent random λ-bit strings will suffice.
5. Compute $\mathbf{C}', e', d' = \mathsf{Gb}\left(1^\lambda, C'\right)$ such that $e' = \left(\left(k_L^0, k_L^1\right), \left(k_R^0, k_R^1\right)\right)$
6. Set $\mathbf{C}^\star = \mathbf{C}||\mathbf{C}'$, and initialize $e^\star = e$ and $d^\star = d||d'$.
7. The encoding and decoding information for $\mathbf{C}^\star$ are made consistent:
   - **if** L does not touch C **then** Update $e^\star = e^\star||e'[0]$.
   - **if** R does not touch C **then** Update $e^\star = e^\star||e'[1]$.
   - For $w \in \{L, R\}$ such that w is the i^{th} output wire of C, update $d^\star = d^\star \setminus d[i]$
8. **return** $\mathbf{C}^\star, e^\star, d^\star$

[a] This can be done by saving the required keys from Step 2

Fig. 10: Complete Specification of a Gate-by-gate Composing Gb Routine

Given garbling scheme $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$, we construct composed garbling scheme $\mathsf{Garble}^\star = (\mathsf{Gb}^\star, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ as per Fig. 10. Garble is composable if there exists a $\mathsf{Garble}^\star$ as per Fig. 10 which achieves privacy. Note that when the output distribution of $\mathsf{Gb}^\star$ is identical to that of Gb, $\mathsf{Garble}^\star$ is equivalent to Garble, and hence Gb can be composed recursively to garble any poly-size circuit gate by gate. Most practical garbling schemes [ZRE15,GLNP15] already follow this template.

Acknowledgements. Part of this work was done while the second author was at IIIT-Bangalore, where he was supported by the Infosys foundation. The authors would like to thank the anonymous reviewers for their feedback.

References

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104, 2017.

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

AIKW15. Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM J. Comput.*, 44(2):433–466, 2015.

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001.

AMPR14. Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.

BCPW15. Fabrice Benhamouda, Geoffroy Couteau, David Pointcheval, and Hoeteck Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 107–129, 2015.

Bea96a. Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 629–638, 1996.

Bea96b. Donald Beaver. Equivocable oblivious transfer. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 119–130, 1996.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

BHR12a. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

BHR12b. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796. ACM, 2012.

BP12. Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 190–208, 2012.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CDG+17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer* *and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842, 2017.

CFGN96. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648, 1996.

CFH+15. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society Press, May 2015.

CGM16. Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530. Springer, Heidelberg, August 2016.

CGP15. Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. *Adaptively Secure Two-Party Computation from Indistinguishability Obfuscation*, pages 557–585. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 597–608. ACM Press, November 2014.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503, 2002.

CM99. Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122. Springer, Heidelberg, May 1999.

CP16. Ran Cohen and Chris Peikert. On adaptively secure multiparty computation with a short crs. In *Proceedings of the 10th International Conference on Security and Cryptography for Networks - Volume 9841*, pages 129–146, 2016.

CPV17. Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. Equivocating yao: constant-round adaptively secure multiparty computation in the plain model. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 497–509, 2017.

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

DI06. Ivan Damgrard and Yuval Ishai. Scalable secure multiparty computation. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 501–520, 2006.

DSKR15. Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. *Adaptively Secure, Universally Composable, Multiparty Computation in Constant Rounds*, pages 586–613. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

FNO15. Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *Advances in Cryptology-EUROCRYPT 2015*, pages 191–219. Springer, 2015.

GG14. Juan A. Garay and Rosario Gennaro, editors. *CRYPTO 2014, Part II*, volume 8617 of *LNCS*. Springer, Heidelberg, August 2014.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

GIKM98. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *30th ACM STOC*, pages 151–160. ACM Press, May 1998.

GK96. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

GLNP15. Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 567–578, 2015.

GMO16. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 1069–1083, 2016.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 171–185, 1986.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GMW91. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

GQ88. Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, Heidelberg, May 1988.

Gro10. Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 341–358. Springer, Heidelberg, December 2010.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 415–432, 2008.

HMR15. Zhangxiang Hu, Payman Mohassel, and Mike Rosulek. Efficient zero-knowledge proofs of non-algebraic statements with sublinear amortized cost. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 150–169, 2015.

HPV17. Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Constant round adaptively secure protocols in the tamper-proof hardware model. In *Proceedings, Part II, of the 20th IACR International Conference on Public-Key Cryptography — PKC 2017 - Volume 10175*, pages 428–460, 2017.

HV16. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 397–429. Springer, Heidelberg, August 2016.

IEE13. *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013.

IKO+11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 406–425, 2011.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

IKOS09. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

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

JKO13. Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 955–966. ACM, 2013.

Kil88. Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

KKL+16. Vladimir Kolesnikov, Hugo Krawczyk, Yehuda Lindell, Alex J. Malozemoff, and Tal Rabin. Attribute-based key exchange with general policies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1451–1463, 2016.

KMO89. Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 474–479, 1989.

KMR14. Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. Flexor: Flexible garbling for XOR gates that beats free-xor. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 440–457, 2014.

KP17. Yashvanth Kondi and Arpita Patra. Privacy-free garbled circuits for formulas: Size zero and information-theoretic. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 188–222, 2017.

KS08. Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgrard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Heidelberg, July 2008.

Lin11. Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 446–466, 2011.

Lin13. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 1–17. Springer, Heidelberg, August 2013.

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.

LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

LR14. Yehuda Lindell and Ben Riva. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. In Garay and Gennaro [GG14], pages 476–494.

LR15. Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 579–590, 2015.

LZ11. Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *J. Cryptology*, 24(4):761–799, 2011.

MPs16. Tal Malkin, Valerio Pastro, and abhi shelat. An algebraic approach to garbling, 2016. Unpublished manuscript, see `https://simons.berkeley.edu/talks/tal-malkin-2015-06-10`.

MR17. Payman Mohassel and Mike Rosulek. Non-interactive secure 2pc in the offline/online and batch settings. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 425–455, 2017.

NOVY98. Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998.

NP05. Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, January 2005.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In IEEE S&P 2013 [IEE13], pages 238–252.

PSSW09. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, Heidelberg, December 2009.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

Rab05. Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. `http://eprint.iacr.org/2005/187`.

RR16. Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 297–314, 2016.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

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

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *Advances in Cryptology-EUROCRYPT 2015*, pages 220–250. Springer, 2015.

A Properties of Garbling Schemes

Definition A.1 *(Correctness) A garbling scheme* Garble *is* correct *if for all input lengths* $n \leq \mathsf{poly}(\lambda)$*, circuits* $C : \{0,1\}^n \to \{0,1\}^m$ *and inputs* $x \in \{0,1\}^n$*, the following probability is negligible in* λ*:*

$$\Pr\left(\mathsf{De}(\mathsf{Ev}(\mathbf{C}, \mathsf{En}(e,x)), d) \neq C(x) : (\mathbf{C}, e, d) \leftarrow \mathsf{Gb}(1^\lambda, C)\right).$$

Definition A.2 *(Privacy) A garbling scheme* Garble *is* private *if for all input lengths* $n \leq \mathsf{poly}(\lambda)$*, circuits* $C : \{0,1\}^n \to \{0,1\}^m$*, there exists a* PPT *simulator* $\mathcal{S}$ *such that for all inputs* $x \in \{0,1\}^n$*, for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, the following two distributions are computationally indistinguishable:*

- $\textsc{Real}(C, x)$: *run* $(\mathbf{C}, e, d) \leftarrow \mathsf{Gb}(1^\lambda, C)$*, and output* $(\mathbf{C}, \mathsf{En}(x, e), d)$*.*
- $\textsc{Ideal}_{\mathcal{S}}(C, C(x))$*: output* $(\mathbf{C}', \mathbf{X}, d') \leftarrow \mathcal{S}(1^\lambda, C, C(x))$

Definition A.3 *(Authenticity) A garbling scheme* Garble *is* authentic *if for all input lengths* $n \leq \mathsf{poly}(\lambda)$*, circuits* $C : \{0,1\}^n \to \{0,1\}^m$*, inputs* $x \in \{0,1\}^n$*, and all probabilistic polynomial-time adversaries* $\mathcal{A}$*, the following probability is negligible in* λ*:*

$$\Pr\begin{pmatrix} \widehat{\mathbf{Y}} \neq \mathsf{Ev}(\mathbf{C}, \mathbf{X}) & \mathbf{X} = \mathsf{En}(x, e),\ (\mathbf{C}, e, d) \leftarrow \mathsf{Gb}(1^\lambda, C) \\ \wedge \mathsf{De}(\widehat{\mathbf{Y}}, d) \neq \bot & : \quad \widehat{\mathbf{Y}} \leftarrow \mathcal{A}(C, x, \mathbf{C}, \mathbf{X}) \end{pmatrix}.$$

Definition A.4 *(Verifiability I) A garbling scheme* Garble *is* verifiable *if for all input lengths* $n \leq \mathsf{poly}(\lambda)$*, circuits* $C : \{0,1\}^n \to \{0,1\}^m$*, inputs* $x \in \{0,1\}^n$*, and PPT adversaries* $\mathcal{A}$*, the following probability is negligible in* λ*:*

$$\Pr\left(\mathsf{De}\left(\mathsf{Ev}(\mathbf{C}, \mathsf{En}(x, e)), d\right) \neq C(x) \ : \ \begin{matrix} (\mathbf{C}, e, d) \leftarrow \mathcal{A}(1^\lambda, C) \\ \mathsf{Ve}\left(C, \mathbf{C}, e, d\right) = 1 \end{matrix}\right)$$

For completeness, we also require the following property of a verifiable garbling scheme:

$$\forall \left(\mathbf{C}, e, d\right) \leftarrow \mathsf{Gb}\left(1^\lambda, C\right),\ \mathsf{Ve}\left(C, \mathbf{C}, e, d\right) = 1$$

Definition A.5 *(Verifiability II) A garbling scheme* Garble *is* verifiable *if there exist a pair of poly-time algorithm* $(\mathsf{Ve}_1, \mathsf{Ve}_2)$ *such that for all input lengths* $n \leq \mathsf{poly}(\lambda)$*, circuits* $C : \{0,1\}^n \to \{0,1\}^m$*, inputs* $x \in \{0,1\}^n$*, and PPT adversaries* $\mathcal{A}$*, the following probability is 1:*

$$\Pr\left(\mathsf{Ve}_1\left(\mathbf{C}, e, C(x)\right) = \mathsf{Ev}\left(\mathbf{C}, \mathsf{En}(x, e)\right) \ : \ \begin{matrix} (\mathbf{C}, e, d) \leftarrow \mathcal{A}(1^\lambda, C) \\ \mathsf{Ve}_2\left(C, \mathbf{C}, e\right) = 1 \end{matrix}\right)$$

For completeness, we also require the following property of a verifiable garbling scheme:

$$\forall \left(\mathbf{C}, e, d\right) \leftarrow \mathsf{Gb}\left(1^\lambda, C\right),\ \mathsf{Ve}_2\left(C, \mathbf{C}, e\right) = 1$$

The notions of verifiability in Definitions A.4 and A.5 can be shown to be separate. Observe that the Ve_2 routine does not even take d as an input, therefore a given (C, e, d) which does not evaluate and decode to the correct output can still be verified positively as per the Verifiability II definition; the task of Ve_2 is only to verify that the garbled output does not reveal any information about the clear input. This definition is insufficient for protocols that rely on the correctness of the GC for correctness of a computation.

On the other hand, consider a trivial garbling scheme in which an evaluator outputs the XOR of her input keys. For instance, garbling an AND gate would produce keys L_0, L_1 on the left incoming wire and R_0, R_1 on the right incoming wire, with the published decoding information being $d = L_1 \oplus R_1$. Decoding a garbled output $Y = L \oplus R$ is just comparing Y to d. This scheme is private and verifiable as per Def. A.4, however it does not achieve Verifiability II; A GC constructor will know exactly which clear input would yield a given garbled output.

We observe that Verifiability II is implicit in the argument of [ZRE15] when proving that a linear garbling scheme produces at least two ciphertexts when garbling an AND gate; that the garbled output depends only on the corresponding clear output is a central assumption.

B The Universal Composability (UC) Security Model

We prove security of our protocol in the standard Universal Composability (UC) framework of Canetti [Can01], with both static and adaptive corruption. The UC framework introduces a PPT environment $\mathcal{Z}$ that is invoked on the security parameter λ and an auxiliary input z and oversees the execution of a protocol in one of the two worlds. The "ideal" world execution involves dummy parties P_0 and P_1, an ideal adversary $\mathcal{S}$, and a functionality $\mathcal{F}$. The "real" world execution involves the PPT parties P_0 and P_1 and a real world adversary $\mathcal{A}$. In the static setting, the ideal adversary $\mathcal{S}$ and the real adversary $\mathcal{A}$ may corrupt one of the parties statically (at the outset). In the adaptive setting, they corrupt the parties adaptively and both parties can be corrupted. The environment $\mathcal{Z}$ chooses the input of the parties and may interact with the ideal/real adversary during the execution. At the end of the execution, it has to decide upon and output whether a real or an ideal world execution has taken place.

We let $\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda, z)$ denote the random variable describing the output of the environment $\mathcal{Z}$ after interacting with the ideal execution with adversary $\mathcal{S}$, the functionality $\mathcal{F}$, on the security parameter 1^λ and z. Let $\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the ensemble $\{\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^*}$. Similarly let $\textsc{real}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\lambda, z)$ denote the random variable describing the output of the environment $\mathcal{Z}$ after interacting in a real execution of a protocol Π with adversary $\mathcal{A}$, the parties, on the security parameter 1^λ and z. Let $\textsc{real}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\textsc{real}_{\Pi,\mathcal{A},\mathcal{Z}}(1^\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^*}$.

Definition B.1 *For $n \in \mathbb{N}$, let $\mathcal{F}$ be a functionality and let Π be an 2-party protocol. We say that Π securely realizes $\mathcal{F}$ if for every PPT real world adversary*

$\mathcal{A}$, there exists a PPT ideal world adversary $\mathcal{S}$, corrupting the same parties, such that the following two distributions are computationally indistinguishable:

$$\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}.$$

The $\mathcal{F}$-hybrid model. In order to construct some of our protocols, we will use secure two-party protocols as subprotocols. The standard way of doing this is to work in a "*hybrid model*" where both the parties interact with each other (as in the real model) in the outer protocol and use ideal functionality calls (as in the ideal model) for the subprotocols. Specifically, when constructing a protocol Π that uses a subprotocol for securely computing some functionality $\mathcal{F}$, the parties run Π and use "ideal calls" to $\mathcal{F}$ (instead of running the subprotocols implementing $\mathcal{F}$). The execution of Π that invokes $\mathcal{F}$ every time it requires to execute the subprotocol implementing $\mathcal{F}$ is called the *$\mathcal{F}$-hybrid execution of Π* and is denoted as $\Pi^{\mathcal{F}}$. The hybrid ensemble $\text{HYB}_{\Pi^{\mathcal{F}},\mathcal{A},\mathcal{Z}}(1^\lambda, z)$ describes $\mathcal{Z}$'s output after interacting with $\mathcal{A}$ and the parties P_0, P_1 running protocol $\Pi^{\mathcal{F}}$. By UC definition, the hybrid ensemble should be indistinguishable from the real ensemble with respect to protocol Π where the calls to $\mathcal{F}$ are instantiated with a realization of $\mathcal{F}$.

B.1 Functionalities

Oblivious Transfer Oblivious transfer (OT) [NP05,Kil88,Rab05] is a protocol between a sender (S) and a receiver (R). In a 1-out-of-2 OT, the sender holds two inputs $a_0, a_1 \in \{0,1\}^k$ and the receiver holds a choice bit σ. At the end of the protocol, the receiver obtains a_σ. The sender learns nothing about the choice bit, and the receiver learns nothing about the sender's other input. The ideal OT functionality is recalled below in Figure 11.

$\mathcal{F}_{\mathsf{OT}}$

Choose: On input $(\texttt{rec}, \text{sid}, b)$ from the R, with $\sigma \in \{0,1\}$, if no messages of the form $(\texttt{rec}, \text{sid}, \sigma)$ is present in memory, store $(\texttt{rec}, \text{sid}, \sigma)$ and send $(\texttt{rec}, \text{sid})$ to S.

Transfer: On input $(\texttt{sen}, \text{sid}, a_0, a_1)$ from S, with $a_0, a_1 \in \{0,1\}^k$, if no messages of the form $(\texttt{sen}, \text{sid}, a_0, a_1)$ is present in memory and a message of the form $(\texttt{rec}, \text{sid}, \sigma)$ is stored, send $(\texttt{sent}, \text{sid}, a_\sigma)$ to R.

Fig. 11: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer

Zero Knowledge A Zero-knowledge (ZK) proof allows a prover to convince a verifier of the validity of a statement, without revealing any other information beyond that. Let R be an NP relation, and $\mathcal{L}$ be the associated language. $\mathcal{L} = \{z \mid \exists x : R(z,x) = 1\}$. A zero-knowledge proof for $\mathcal{L}$ lets the prover convince a verifier that $z \in \mathcal{L}$ for a common input z. A proof of knowledge captures not only the truth of a statement $z \in \mathcal{L}$, but also that the prover "possesses" a witness x to this fact. A proof of knowledge for a relation $R(\cdot, \cdot)$ is an interactive protocol where a prover P convinces a verifier V that P knows a x such that $R(z, x) = 1$, where z is a common input to P and V . The prover can always successfully convince the verifier if indeed P knows such a x. Conversely, if P can convince the verifier with high probability, then he "knows" such a x, that is, such a x can be efficiently computed given z and the code of P. When the soundness holds only for a PPT prover, it is called an *argument*. As in [JKO13], we define the ideal functionality for zero-knowledge $\mathcal{F}_{\mathsf{ZK}}^{R}$ in the framework of [Can01] in order to capture all the properties that we require, in Figure 12.

$\mathcal{F}_{\mathsf{ZK}}^{R}$

1. Receive $(\texttt{prove}, \mathrm{sid}, z, x)$ from P and $(\texttt{verify}, \mathrm{sid}, z')$ from V
2. **if** $z = z'$ and $R(z, x) = 1$ **then** output $(\texttt{verified}, \mathrm{sid}, z)$ to V

Fig. 12: The Zero-knowledge functionality

Committed OT and Commitment Functionalities The $\mathcal{F}_{\mathsf{COT}}$ and $\mathcal{F}_{\mathsf{COM}}$ functionalities are provided in Fig. 13 and Fig. 14 respectively. The $\mathcal{F}_{\mathsf{COT}}$ functionality can be securely realised in the framework of [PVW08] with an augmentation for the **Open-all** property, as discussed in [JKO13]. The $\mathcal{F}_{\mathsf{COM}}$ functionality can be securely and efficiently realised as well [Lin11].

This is the ideal functionality for Committing Oblivious Transfer, borrowed from [JKO13]. A Sender S provides two messages, of which a Receiver R chooses to receive one. S doesn't know which message R chose, and R has no information about the message it didn't choose. Upon receiving a signal from S, the functionality reveals both messages to R.

$\mathcal{F}_{\mathsf{COT}}$

1. **Choose:** Receive $(\texttt{choose}, id, b)$ from R, where $b \in \{0, 1\}$. If no message of the form $(\texttt{choose}, id, \cdot)$ exists in memory, store $(\texttt{choose}, id, b)$ and send $(\texttt{chosen}, id)$ to S.
2. **Transfer:** Receive $(\texttt{transfer}, id, tid, m_0, m_1)$ from S, where $m_0, m_1 \in \{0, 1\}^{\kappa}$. If no message of the form $(\texttt{transfer}, id, tid, \cdot, \cdot)$ exists in memory, and a message of the form $(\texttt{choose}, id, b)$ exists in memory, then send $(\texttt{transferred}, id, tid, m_b)$ to R.
3. **Open-all:** Receive $(\texttt{open-all})$ from the S. Send all messages of the form $(\texttt{transfer}, id, tid, m_0, m_1)$ to R.

Fig. 13: The Ideal Committing OT functionality

The ideal commitment functionality, borrowed from [JKO13]. A Sender S commits to a message m, which she later reveals to the receiver R. S is 'bound' to only the message that she committed, while the message is hidden from R until S opens her commitment.

$\mathcal{F}_{\mathsf{COM}}$

1. **Commit:** Receive $(\texttt{commit}, id, m)$ from the sender, where $m \in \{0,1\}^*$. If no such message already exists in memory, then store $(\texttt{commit}, id, m)$ and send $(\texttt{committed}, id, |m|)$ to R.
2. **Reveal:** Receive $(\texttt{reveal}, id)$ from S, send $(\texttt{reveal}, id, m)$ to R if corresponding $(\texttt{commit}, id, m)$ exists in memory.

Fig. 14: The Ideal Commitment Functionality

C Schematic Diagrams

D Proof of Security for 2-Round ZK

Theorem D.1 *Let* $\mathsf{Garble} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{Ve})$ *be a correct, private, verifiable (according to Definition A.4) garbling scheme. Then protocol* Π_{ZK2} *securely realizes ideal functionality* $\mathcal{F}^R_{\mathsf{ZK}}$ *in the presence of malicious adversaries in the* $\mathcal{F}_{OT}$*-hybrid model.*

Proof. Our proof is presented in Universal Composability (UC) framework recalled briefly in Appendix B. To prove the security of our protocol, we describe two simulators. The simulator $\mathcal{S}_{\mathsf{P}}$ simulates the view of a corrupt prover and appears in Fig. 18. On the other hand, the simulator $\mathcal{S}_{\mathsf{V}}$ simulates the view of a corrupt verifier and is presented in Fig. 19.

Security against a Corrupt Prover $\mathsf{P}^\star$. We now prove that $\textsc{ideal}_{\mathcal{F}^R_{\mathsf{ZK}}, \mathcal{S}_{\mathsf{P}}, \mathcal{Z}} \stackrel{c}{\approx} \textsc{real}_{\Pi_{\mathsf{ZK2}}, \mathcal{A}, \mathcal{Z}}$ when $\mathcal{A}$ corrupts P. The simulation deviates from the real world if and only if every $\mathbf{C}$ evaluated by V is 'bad', and every $\mathbf{C}$ checked by V is valid. In this case $\mathsf{P}^\star$ succeeds in cheating, i.e. V outputs accept in the real world, whereas the simulator outputs reject. Since we are in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, the association of a circuit being bad or valid is fixed before the challenge is revealed, and a $\mathsf{P}^\star$ attempting to cheat in this way would have to correctly guess the entire μ-bit challenge string of V. This occurs with probability $2^{-\mu}$.

Security against a Corrupt Verifier $\mathsf{V}^\star$. We now argue that $\textsc{ideal}_{\mathcal{F}^R_{\mathsf{ZK}}, \mathcal{S}_{\mathsf{V}}, \mathcal{Z}} \stackrel{c}{\approx} \textsc{real}_{\Pi_{\mathsf{ZK2}}, \mathcal{A}, \mathcal{Z}}$ when $\mathcal{A}$ corrupts V. The above two views of V^* are shown to be indistinguishable via a series of intermediate hybrids. The difference between the two world lies in the way the evaluation garbled circuits are constructed. In the real world, they are constructed using Gb, whereas in the simulated world, they are created using the privacy simulator of the garbling scheme. Assuming that there are β c_i bits are 1, the indistinguishability between the simulated and the real view can be shown through β $(\leq \mu)$ hybrids where $\textsc{hyb}_0$ is same as

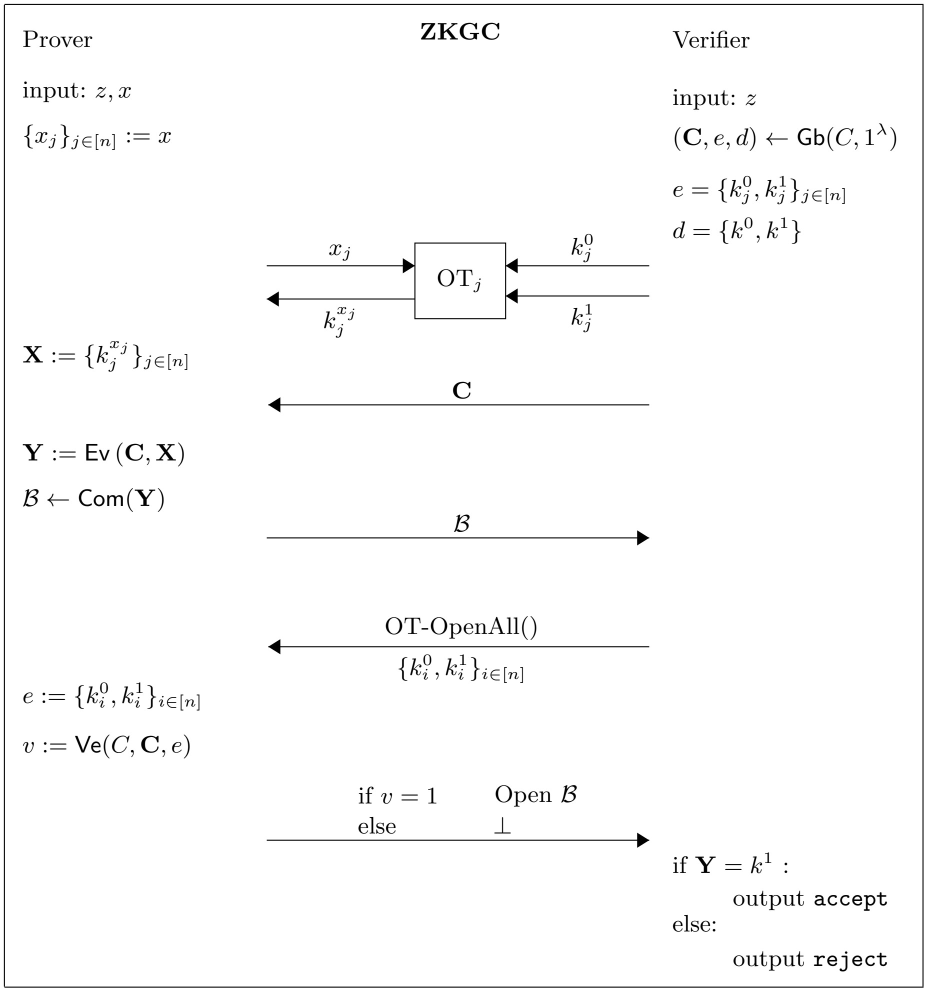

Fig. 15: ZKGC: Zero-knowledge from one GC [JKO13]

$\text{REAL}_{\Pi_{\mathsf{ZK2}},\mathcal{A},\mathcal{Z}}$ and HYB_j is same as HYB_{j-1} except that jth evaluation circuit is constructed using the privacy simulator of the garbling scheme. Clearly, HYB_β is same as the simulated view and indistinguishability of two consecutive follows from the privacy of the underlying garbling scheme. Namely, privacy by Definition A.2 implies there exists a PPT simulator $\mathcal{S}_{\mathsf{Garble}}$ such that for all (C, x), the distribution of $(\mathbf{C}, \mathbf{X}, d) = \mathcal{S}_{\mathsf{Garble}}(1^\lambda, C, C(x))$ is computationally indistinguishable from $(\mathbf{C}', \mathbf{X}', d')$ where $(\mathbf{C}', e', d') \leftarrow \mathsf{Gb}(1^\lambda, C)$, $\mathbf{X}' = \mathsf{En}(x, e')$.

□

3-round Zero-Knowledge protocol Π_{ZK3}

Prover P | Verifier V

Prover P

input: z, x

$\{r_j^{\mathsf{R}}\}_{j=1}^{n} \leftarrow \{0,1\}^{\lambda}$

$r^{\mathsf{R}} = r_1^{\mathsf{R}} || \ldots || r_n^{\mathsf{R}}$

$\{m_j^{\mathsf{R}}\}_{j\in[n]} = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, x_j; r_j^{\mathsf{R}})$

Verifier V

input: z

$(\mathbf{C}, e, d) \leftarrow \mathsf{Gb}\left(C, 1^{\lambda}\right)$

$e = \{k_j^0, k_j^1\}_{j\in[n]}, d = \{k^0, k^1\}$

$\mathsf{P} \rightarrow \mathsf{V}$: $\{m_j^{\mathsf{R}}\}_{j\in[n]}$

Verifier V

$\{r_j^{\mathsf{S}}\}_{j\in[n]} \leftarrow \{0,1\}^{\lambda}$

$r^{\mathsf{S}} = r_1^{\mathsf{S}} || \ldots || r_n^{\mathsf{S}}$

$\{m_j^{\mathsf{S}}\}_{j\in[n]} = \Pi_{\mathsf{OT}}^{\mathsf{S}}(\mathsf{crs}, k_j^0, k_j^1, m_j^{\mathsf{R}}; r_j^{\mathsf{S}})$

$\mathsf{V} \rightarrow \mathsf{P}$: $\mathbf{C}, T = H(k^1) \oplus r^{\mathsf{S}}, \{m_j^{\mathsf{S}}\}_{j\in[n]}$

Prover P

$k_j^{x_j} = \Pi_{\mathsf{OT}}^{\mathsf{R}}(\mathsf{crs}, x_j, m_j^{\mathsf{S}}; r_j^{\mathsf{R}})_{j\in[n]}$

$\mathbf{Y} = \mathsf{Ev}(\mathbf{C}, \{k_j^{x_j}\}_{j\in[n]})$

$r^{\mathsf{S}} = H(\mathbf{Y}) \oplus T$

$r^{\mathsf{S}} = r_1^{\mathsf{S}} || \ldots || r_n^{\mathsf{S}}$

if $\exists j, \mathsf{Ext}(\mathsf{crs}, m_j^{\mathsf{R}}, m_j^{\mathsf{S}}, r_j^{\mathsf{S}}) = \perp$, then abort

else $(k_j^0, k_j^1) = \mathsf{Ext}(\mathsf{crs}, m_j^{\mathsf{R}}, m_j^{\mathsf{S}}, r_j^{\mathsf{S}}), \forall j \in [n]$

if $\mathsf{Ve}_2(C, \mathbf{C}, \{k_j^0, k_j^1\}_{j\in[n]}) = 0$, then abort

$\mathsf{P} \rightarrow \mathsf{V}$: $\mathbf{Y}$

Verifier V

if $(\mathbf{Y} = k^1)$:

output `accept`

else output `reject`

Fig. 16: Schematic diagram of our 3 round Zero-Knowledge protocol Π_{ZK3}

2-round protocol for Zero-Knowledge Proof Π_{ZKP}

Prover P	Verifier V
input: z, x	input: z
$i = 1$ to μ :	
$(\mathbf{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\lambda, C)$	
$\mathbf{X}_i = \mathsf{En}(x, e_i)$	
for $j \in [n]$:	
$b_{ij} \leftarrow \{0,1\}$	$c \leftarrow \{0,1\}^\mu$
$(\mathcal{B}_{ij}^0, \mathcal{B}_{ij}^1) = (\mathsf{Com}(\mathbf{X}_{ij}^{b_{ij}}), \mathsf{Com}(\mathbf{X}_{ij}^{1-b_{ij}}))$	$i = 1$ to μ :
(e_i, r_i) → OT_i	← c_i
$(\mathbf{X}_i, s_i)$ → OT_i	→ (e_i, r_i), if $c_i = 0$; $(\mathbf{X}_i, s_i)$, otherwise
$\mathbf{C}_i, d_i, (\mathcal{B}_{ij}^0, \mathcal{B}_{ij}^1)_{j \in [n]}$ →	

if $c_i = 0$:
if $\mathsf{Ve}(C, \mathbf{C}_i, e_i) = 0$ or
$\exists j \in [n], (e_i, r_i)$ is inconsistent
with $(\mathcal{B}_{ij}^0, \mathcal{B}_{ij}^1)$
outputs reject and halt

if $c_i = 1$:
if $\exists j \in [n], (\mathbf{X}_i, s_i)$ inconsistent
with $\mathcal{B}_{ij}^{b_{ij}}$ for some b_{ij}
or $\mathsf{De}(\mathsf{Ev}(\mathbf{C}_i, \mathbf{X}_i), d_i) \neq 1$
outputs reject and halt
outputs accept

Fig. 17: Schematic diagram of our 2 round protocol for Zero-Knowledge Proof Π_{ZKP}

Simulator $\mathcal{S}_{\mathsf{P}}$

The simulator plays the role of the honest V and simulates each step of the protocol Π_{ZK3} as follows. The communication of the $\mathcal{Z}$ with the adversary $\mathcal{A}$ who corrupts P is handled as follows: Every input value received by the simulator from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to the simulator's output tape (to be read by the environment $\mathcal{Z}$).

OT First Message Phase: On behalf on $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{S}_{\mathsf{P}}$ sends $(\texttt{rec}, \mathrm{sid})$ to P.

OT Second Message and Circuit Communication Phase: For all $i \in [\mu]$, the simulator $\mathcal{S}_{\mathsf{P}}$ receives $(\mathbf{C}_i, d_i)$ from $\mathsf{P}^\star$, and $(\texttt{sen}, \mathrm{sid}, e_i, \mathbf{X}_i)$ on behalf of $\mathcal{F}_{\mathsf{OT}}$. Given e_i as well as $\mathbf{X}_i$, $\mathcal{S}_{\mathsf{P}}$ extracts x_i such that $\mathsf{En}(x_i, e_i) = \mathbf{X}_i$ for every $i \in [\mu]$. For instance in the case of projective garbling schemes, this can be done by comparing the j^{th} encoded input token $\mathbf{X}_{ij}$ with $e_{ij} = (k_{ij}^0, k_{ij}^1)$ to extract the bit x_{ij}, $\forall j \in [n]$.

Circuit Checking, Evaluation and Output Phase: If $\exists x \in \{x_i\}_{i\in[\mu]}$ that is valid, i.e. $C(z, x) = 1$, then $\mathcal{S}_{\mathsf{P}}$ behaves exactly like an honest verifier, and outputs `accept` or `reject` based on the same criteria that a real world V would follow. However, if $\forall x \in (x_i)_{i\in[\mu]}$ such that $C(z, x) \neq 1$ then it outputs `reject`.

Fig. 18: Simulator $\mathcal{S}_{\mathsf{P}}$

Simulator $\mathcal{S}_{\mathsf{V}}$

The simulator plays the role of the honest P and simulates each step of the protocol Π_{ZK3} as follows. The communication of the $\mathcal{Z}$ with the adversary $\mathcal{A}$ who corrupts V is handled as follows: Every input value received by the simulator from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to the simulator's output tape (to be read by the environment $\mathcal{Z}$).

OT First Message Phase: On behalf on $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{S}_{\mathsf{V}}$ receives $(\texttt{rec}, \mathrm{sid}, c_i)$.

OT Second Message and Circuit Communication Phase: For all $i \in [\mu]$, $\mathcal{S}_{\mathsf{V}}$ proceeds as follows:

- If $c_i = 0$ then $\mathcal{S}$ constructs $(\mathbf{C}_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\lambda, C)$ and sends $(\texttt{sent}, \mathrm{sid}, e_i)$ to $\mathsf{V}^\star$ on behalf of $\mathcal{F}_{\mathsf{OT}}$. $\mathcal{S}_{\mathsf{V}}$ also sends $(\mathbf{C}_i, d_i)$ to $\mathsf{V}^\star$.
- Else $\mathcal{S}_{\mathsf{V}}$ constructs $(\mathbf{C}_i, \mathbf{X}_i, d_i) \leftarrow \mathcal{S}_{\mathsf{Garble}}(1^\lambda, C, 1)$ and sends $(\texttt{sent}, \mathrm{sid}, \mathbf{X}_i)$ to $\mathsf{V}^\star$ on behalf of $\mathcal{F}_{\mathsf{OT}}$ where $\mathcal{S}_{\mathsf{Garble}}$ is the privacy simulator of the garbling scheme. $\mathcal{S}_{\mathsf{V}}$ also sends $(\mathbf{C}_i, d_i)$ to $\mathsf{V}^\star$.

Circuit Checking, Evaluation and Output Phase: $\mathcal{S}_{\mathsf{V}}$ does nothing.

Fig. 19: Simulator $\mathcal{S}_{\mathsf{V}}$